Rumour Stance and Veracity Classification in Social Media Conversations

by

Elena Kochkina

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Department of Computer Science

Contents

List of Tables

List of Figures

Acknowledgments

Throughout a PhD, with its ups and downs, I have always felt a supporting net of people around: my supervisor, colleagues, friends and family. Here I thank them.

I am immensely grateful to my main supervisor Maria Liakata for her support and advise. Thank you for all-nighters before deadlines, Skypes on the weekends, involving me in events and meetings that provided opportunities for learning and growth. Thank you for being kind, calm, confident and honest. You taught me to believe in my work (though I am still working on it). I am very fortunate to have you as my mentor.

During my PhD I was incredibly lucky to meet and collaborate with Arkaitz Zubiaga, Isabelle Augenstein, Genevieve Gorell, Rob Procter, Leon Derczynski, Harish Tayyar Madabushi, Michael Castelle and many other brilliant people, who always showed their passion for the subject, which pushed me to strive for more. A special mention goes to Adam Tsakalidis, who was always available for my questions and also for big slabs of feta on lunches he made for me when it was tough before deadlines. Also, to my friends Drs. Victoria Ponce, Merve Alanyali and Chanuki Seresinhe, who are the role-models of bright, strong women conquering the world.

A huge thank you to the Bridges Programme and Warwick Computer Science department for selecting me and awarding me a scholarship that made all of this possible. I am also grateful to the Alan Turing Institute for its enrichment program and being an amazing place for research and collaboration.

I thank my examiners Andreas Vlachos and Yulan He for taking the time to read this so very thoroughly.

Finally, this thesis would not have been possible without the support of my family. Mum, dad, even though I missed home being away from you, PhD was an adventure worth taking. Thank you to my partner's mum and auntie for their warmth, laughter and ginger. And, most of all, I thank and praise my partner Jason for being my strength, courage and optimism.

Declarations

This thesis is submitted to the University of Warwick in support of my application for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree.

The work presented was carried out primarily by the author except in the cases outlined below:

- Work in section 4.7.3 has been done in collaboration with Arkaitz Zubiaga, Maria Liakata, Rob Procter, Michal Lukasik, Kalina Bontcheva, Trevor Cohn and Isabelle Augenstein. The author of this thesis have performed development of branch-LSTM model and testing of different feature sets with branch-LSTM model, while results of testing feature sets with other models were provided by collaborators. It is important to present results and conclusions of this collaborative effort in this thesis as the insights obtained guided further experiments by the author.

Parts of this thesis have been published as full papers by the author:

- **Kochkina, E.**, Liakata, M., & Augenstein, I. (2017, August). Turing at SemEval-2017 Task 8: Sequential Approach to Rumour Stance Classification with Branch-LSTM. In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017) (pp. 475-480) Kochkina et al. (2017).

- Zubiaga, A., **Kochkina, E.**, Liakata, M., Procter, R., Lukasik, M., Bontcheva, K., & Augenstein, I. (2018, December). Discourse-aware rumour stance classification in social media using sequential classifiers. Information Processing & Management, 54(2), 273-290 Zubiaga et al. (2018a).

- **Kochkina, E.**, Liakata, M., & Zubiaga, A. (2018, August). All-in-one: Multi-task Learning for Rumour Verification. In Proceedings of the 27th International Conference on Computational Linguistics (pp. 3402-3413) Kochkina et al. (2018).

- Gorrell, G., **Kochkina, E.**, Liakata, M., Aker, A., Zubiaga, A., Bontcheva, K., & Derczynski, L. (2019, June). SemEval-2019 task 7: RumourEval, determining rumour veracity and support for rumours. In Proceedings of the 13th International Workshop on Semantic Evaluation (pp. 845-854) Gorrell et al. (2019).

- **Kochkina, E.**, Liakata, M. (2019, September) Incorporating uncertainty estimation into rumour verification. Under submission.

The author has also contributed to the following works, albeit not directly linked to this thesis:

- Zubiaga, A., **Kochkina, E.**, Liakata, M., Procter, R., & Lukasik, M. (2016, December). Stance Classification in Rumours as a Sequential Task Exploiting the Tree Structure of Social Media Conversations. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers (pp. 2438-2448) Zubiaga et al. (2016a).

- Madabushi, HT., **Kochkina, E.**, & Castelle, M. (2019, November). Cost-Sensitive BERT for Generalisable Sentence Classification on Imbalanced Data. To appear in proceedings of 2nd Workshop on NLP for Internet Freedom (NLP4IF): Censorship, Disinformation, and Propaganda.

Abstract

Social media platforms are popular as sources of news, often delivering updates faster than traditional news outlets. The absence of verification of the posted information leads to wide proliferation of misinformation. The effects of propagation of such false information can have far-reaching consequences on society. Traditional manual verification by fact-checking professionals is not scalable to the amount of misinformation being spread. Therefore there is a need for an automated verification tool that would assist the process of rumour resolution. In this thesis we address the problem of rumour verification in social media conversations from a machine learning perspective.

Rumours that attract a lot of scepticism in the form of questions and denials among the responses are more likely to be proven false later (Zhao et al., 2015). Thus we explore how crowd wisdom in the form of the stance of responses towards a rumour can contribute to an automated rumour verification system. We study the ways of determining the stance of each response in a conversation automatically. We focus on the importance of incorporating conversation structure into stance classification models and also identifying characteristics of supporting, denying, questioning and commenting posts. We follow by proposing several models for rumour veracity classification that incorporate different feature sets, including the stance of the responses, attempting to find the set that would lead to the most accurate models across several datasets. We view the rumour resolution process as a sequence of tasks: rumour detection, tracking, stance classification and, finally, rumour verification. We then study relations between the tasks in the rumour verification pipeline through a joint learning approach, showing its benefits comparing to single-task learning. Finally, we address the issue of transparency of model decisions by incorporating uncertainty estimation methods into rumour verification models. We then conclude and point directions for future research.

Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
BOW	Bag-of-Words
CBOW	Continuous Bag-of-Words
DTW	Dynamic Time Wrapping
CRF	Conditional Random Field
EAVI	European Association for Viewers Interests
ELMO	Embeddings from Language Models
EXIF	Exchangeable Image File Format
FF	Feed Forward
FN	False Negative
FP	False Positive
GP	Gaussian Process
GPT	Generative Pre-trained Transformer
GRU	Gated Recurrent Unit
HMM	Hidden Markov Models
ID	IDentification
LIWC	Linguistic Inquiry and Word Count
LM	Language Model
LOEOCV	Leave-One-Event-Out Cross-Validation
LR	Linear/Logistic Regression

LSTM	Long Short-Term Memory
ML	Machine Learning
NLP	Natural Language Processing
NN	Neural Network
PCA	Principal Component Analysis
POS	Part Of Speech
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RF	Random Forest
RMSE	Root Mean Squared Error
RNN	Recurrent/Recursive Neural Network
SG	Skip-Gram
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
UGC	User Generated Content
URL	Uniform Resource Locator

CHAPTER 1

Introduction

1.1 Motivation

Social media platforms have gained tremendous popularity as news sources for their users. According to the Pew Research Center 67% of American adults (ages 18+) were getting news from social media in 2017 (Shearer and Gottfried, 2017). Social media platforms are faster at delivering updates on breaking news than traditional news sources. Further, social media platforms, such as Twitter, can be used by researchers and journalists to detect, geolocate and monitor news events, in particular crisis events such as hurricanes (Hughes and Palen, 2009), floods (Vieweg et al., 2010) and earthquakes (Earle et al., 2010). However, these gains in speed are partially due to the fact that the step of story verification is often omitted and any user can become a 'reporter', giving rise to the spread of misinformation. Rumours tend to rise particularly around situations that cause people to feel uncertainty and anxiety (Allport and Postman, 1965), especially when confirmations from an official source are unavailable (Oh et al., 2013). In this work we are concerned with rumours spreading in social media conversations, which we define as claims related to an event and unverified at the time of posting and circulation, that is verifieable and spreading widely, such that it can be later resolved as either true, false or remain unverified (Zubiaga et al., 2018b).

Often rumourous claims spread on social media rapidly and get exposed to a wide audience before they are verified. The wide spread of false rumours carries a lot of risks, as information from social media is being used by the public and various professionals in their decision-making. Examples include the use by journalists for news gathering, by police and rescuers in cases of emergencies and crisis events, by politicians and policy-makers to understand public opinion and by stock traders to predict stock market reactions. One of the fraud cases that

attracted public attention[1] was of a British stock trader who, in 2013, tweeted false statements about two companies to manipulate share prices, causing a loss of $1.6m to shareholders. An analysis of news leading up to the 2016 US presidential election conducted by BuzzFeed[2], found that there was more engagement with misleading news stories than real news stories. These cases of misinformation affecting society, along with many others, have lead to a rise in public concern about misinformation and disinformation. According to the Reuters Institute Digital News Report 2019, there is high levels of concern among users about which information is real and which is fake on the internet (in Brazil 85%, the UK 70% and US 67% of survey respondents agreed to having this concern) (Newman et al., 2019). The catastrophic potential of misinformation, combined with the scale of social concern of the issue, highlights how crucial it is to address the problem of rumour spread in social media and to prevent the potential damage caused by misinformation.

Traditionally rumour verification and fact-checking are performed manually by professionals: journalists before releasing information into the news, or analysts for financial statements and decision-making. This process typically involves evaluating the provenance of content, identifying source, looking for the inconsistencies in the date and location information. Often results in attempts to contact the original source in order to get confirmation or catch a lie. Even with the recently increased effort from governments (Wright and David, 2019), social media platforms[3,4] and journalists[5], the laborious manual verification process does not scale proportionally to the speed of information spread and the amount of claims to verify. Therefore there is a need for an automated system that could aid the verification process. In this thesis we aim to explore the ways in which Machine Learning and Natural Language Processing[6] can be employed to perform rumour verification, the process of confirming the truthfulness of the claim, in order to prevent further spread and its consequences.

Automated rumour verification system may have some similarities and also crucial differences from the journalist approach. The features that journalists pay attention to, such as the date of profile creation, geolocation of the user, number of re-tweets and followers, can also be incorporated in a rumour classification system.

[1] http://www.independent.co.uk/news/uk/crime/scottish-stock-market-trader-cost-shareholders-1m-with-fake-tweets-a6724821.html, accessed on 20.09.2019.

[2] https://www.buzzfeednews.com/article/craigsilverman/viral-fake-election-news-outperformed-real-news-on-facebook, accessed on 01.09.2019.

[3] https://facebookjournalismproject.com/, accessed on 20.09.2019.

[4] https://newsinitiative.withgoogle.com/, accessed on 20.09.2019

[5] https://credibilitycoalition.org/results/, accessed on 20.09.2019.

[6] The explanation of terminology used in this chapter can be found in the glossary of terms above and in chapter 2.

An automated system would be able to process vast amounts of information faster than a human reader, as it aims to find patterns indicative of rumours, especially false rumours, in already verified stories; specifically patterns that are generalisable and lead to the verification of a new stories.

Rumour verification is a complex issue as even human professionals can be mistaken. Rumour spread concerns a vast variety of topics, hence discussions around these rumours attract the attention of different audiences. In addition, the veracity of a rumour may change over time with developments in real-world events. The Reuters Institute reports the average level of trust in the news, across all countries and platforms included in the study, to be down to 42% with less than half (49%) agreeing that they trust the news media they themselves use (Newman et al., 2019). These aspects pose high requirements on any, to be created, automated system for rumour verification. To name a few, the system should: be accurate; generalisable to unseen rumours; time-sensitive to update predictions over time; real-time to aim for early predictions; provide justification and/or explanations for its predictions; inform humans of its uncertainty; and be impartial towards biases such as source bias. This is therefore an extremely challenging goal.

1.2 Research Questions and Objectives

We have previously discussed that an automated system for rumour verification is highly desirable. However this goal is not easy to achieve as such a system would be very complex and multi-faceted, given that there are a lot of requirements it would need to satisfy. Therefore, we limit ourselves to rumours that arise in Twitter conversation threads. In this work we only focus on a subset of research questions that lead to progress towards that ultimate goal. Here we outline the research questions of this thesis with the corresponding objectives required to answer them.

- **RQ1** Can we predict rumour veracity from social media conversations?
  - **OBJ 1.1** Investigate existing literature in the field of automated rumour verification to gain understanding of reported patterns of rumours on social media platforms, common approaches and gaps in the knowledge.
  - **OBJ 1.2** Identify relevant realistic datasets of conversations discussing rumours containing ground truth labels.
  - **OBJ 1.3** Build veracity classification models using the identified datasets.
  - **OBJ 1.4** Evaluate the models in a realistic scenario to test their ability to be employed as real-world applications.

- **RQ2** Which aspects of a conversation discussing a rumour are helpful for resolving rumour veracity? Are patterns of support and denial in a conversation indicative of its veracity?
  - **OBJ 2.1** Extract features from social media conversations discussing each rumour, including stance of users towards a rumour, and analyse their presence in each of the true, false and unverified classes.
  - **OBJ 2.2** Incorporate features into the classification models and perform comparison of their performance.
  - **OBJ 2.2** Analyse the outcomes of the experiments in order to identify the best performing feature set.
- **RQ3** Can we automatically identify the stance of users towards rumours from the posts in a conversation?
  - **OBJ 3.1** Identify relevant, realistic datasets of conversation discussing rumours with each tweet annotated with their stance towards the rumour.
  - **OBJ 3.2** Build stance classification models using the identified datasets.
  - **OBJ 3.3** Evaluate the models in a realistic scenario to test their ability to be employed as real-world applications.
- **RQ4** Which linguistic and network features are indicative of stance categories?
  - **OBJ 4.1** Extract features from each of the posts in social media conversations discussing a rumour, and analyse the presence of those features in each of the supporting, denying, questioning and commenting classes.
  - **OBJ 4.2** Incorporate features into the stance classification models and perform comparison of their performance.
  - **OBJ 4.3** Analyse the outcomes of the experiments in order to identify the best performing feature set.
- **RQ5** Can we leverage the sequence of responses to improve stance classification models?
  - **OBJ 5.1** Develop accurate and robust models that can process sequential data leveraging information from conversation structure.
  - **OBJ 5.2** Evaluate developed models against baselines not using conversation structure.

- **RQ6** How can we leverage the interaction between the task of rumour verification and other tasks such as stance classification and rumour detection?
  - **OBJ 6.1** Develop accurate and robust models that use the tasks from rumour resolution pipeline: rumour detection, stance and veracity classification, in a joint learning set up in order to utilise the relations between them.
  - **OBJ 6.2** Evaluate developed models against single-task learning baselines.
- **RQ7** How can we define and use model uncertainty in rumour verification? What does the uncertainty of predictions tell us about the task or the dataset?
  - **OBJ 7.1** Incorporate uncertainty estimation methods into a chosen rumour verification model.
  - **OBJ 7.2** Provide quantitative and qualitative insights on the suitability of the proposed approach.

1.3 Technical Challenges

In this thesis we address the complex tasks of rumour stance classification and veracity classification, and the many technical challenges that they present.

- **CH1 Modelling conversation structure.** As one of the research questions we study is the benefit of incorporating conversation structure into models for rumour stance and veracity classification, we face the challenge of finding a way to model social media conversations. We assume that online conversations involving multiple users, replying to each other at different points in time, in general have a tree-like structure that consists of branching sequences of responses to the source post. We then make use of tree-structured conversations from Twitter started by tweets conveying rumours. Standard models, such as Support Vector Machines (SVM) and Random Forest (RF), can only take an input in the form of a vector and do not model the conversation structure directly. We find a way to introduce conversation information into those models through measuring the proportion of stances expressed towards the rumour in the conversation, and by measuring cosine distance between the vector representation of the source post (that is, the root node of a tree) and the representation of other posts. Further, we also use models that are able to

process conversation structure directly, as a linear sequence of tweets (linear Conditional Random Field (CRF) and Long Short-Term Memory (LSTM) recurrent neural networks) and as a whole tree (tree CRF). However, we mainly leave neural modelling of tree structures for future work (see RQ1, RQ3 and chapters 4–7).

- **CH2 Class imbalance.** The majority of real world datasets have a class imbalance, i.e. have larger number of instances of one of the classes compared to the other classes. Compared to the overall amount of discussions in online social media, rumours take a smaller proportion, however it is very important to distinguish them from chat, opinion and genuine news. The conversations discussing rumours tend to attract a lot of comments that do not question the veracity of the rumour, compared to the posts that support, deny or question the rumour. Therefore we are faced with the problem of a class imbalance in both the stance and veracity classification tasks. Evaluation of models on imbalanced sets using accuracy leads to high scores for models which only predict the majority class, ignoring the minority class. In order to mitigate this effect and evaluate models performance with respect to all of the classes with equal weight we choose macro-averaged F-score as our main metric (see RQ1, RQ3 and chapters 4–7).

- **CH3 Domain adaptation.** Rumours arise around a variety of different events and can concern an array of diverse topics. Therefore training a model on past verified rumours and then testing or applying it to new unseen rumours introduces a situation of domain adaptation, where a model has to adapt to a new topic and its vocabulary. This is an inevitable situation in realistic scenario, as we do not know what would be the focus of the next rumour. Thus we attempt to imitate a realistic scenario by using a leave-one-event-out cross-validation evaluation set up, where a model is tested on event(s) unseen during training (see RQ1 and chapters 5–7).

- **CH4 Certainty of model predictions.** In order to trust the decisions of the model one needs to understand them better. Understanding decisions of deep learning models is a complex question. We are using methods for estimating model uncertainty in order to get a greater understanding of the task of rumour verification and the datasets with which we are working (see RQ1, RQ7 and chapter 7).

1.4 Contributions

In this thesis we make the following contributions:

- We show the importance of the use of conversation structure for rumour stance classification in social media conversations (see RQ3, RQ5 and chapter 4).
- We provide an analysis of the effect of incorporating various groups of relevant features into rumour stance classification models. As a result of this analysis we find a feature set that leads to the best performance compared to other feature combinations (see RQ3, RQ4 and chapter 4).
- We provide an analysis of the effect of incorporating various groups of relevant features into rumour veracity classification models, across several models and datasets. We show that it is hard to find a single feature set that would lead to performance improvements across multiple datasets. This finding highlights the complexity of the task and urges researchers in the field to aim for verifying their findings across multiple datasets (see RQ1, RQ2 and chapter 5).
- We propose and justify evaluation of rumour stance classification and rumour verification models in a leave-one-event-out cross-validation set up as it imitates a realistic scenario where the model is tested on events, unseen during training. This set up tests the ability of models to generalise to new events (see RQ1, RQ3 and chapters 4–7).
- We propose a way of leveraging interactions between the tasks in the rumour verification pipeline through multitask learning and show that it leads to results that outperform the single-task learning approach (see RQ1, RQ6 and chapter 6).
- We show that estimating the uncertainty of rumour verification models can improve model performance by removing instances with low confidence from the model and can also provide insights about both the rumour verification task and datasets used (see RQ1, RQ7 and chapter 7).

1.5 Thesis Outline

This Ph.D. thesis has a traditional outline, starting with background information, followed by four analysis chapters and closing with a conclusion. Chapters 1 and 2 provide the motivation and necessary background for the comprehension of this

thesis. Chapter 3 describes the datasets used in the following analysis. Then, Chapters 4–7 describe the analysis performed on the rumour stance classification and veracity classification tasks. Finally we conclude in Chapter 8.

This thesis is structured as follows:

- **Chapter 1**, the current chapter, is an introduction to the thesis containing the motivation to pursue the problem of rumour verification, as well as the relevant research questions, objectives and challenges.
- **Chapter 2** is split into two parts. The first part provides an introduction to the fields of Machine Learning and Natural Language Processing, in addition to the terms and algorithms that will be used throughout this thesis. The second part provides a literature review of the rumour verification field.
- **Chapter 3** describes the datasets that were used in the analysis in this thesis.
- **Chapter 4** focuses on the task of rumour stance classification. In this chapter we present the hypothesis of its sequential nature, experiments testing it and their outcomes. We also explore relevant features that improve the performance of models for automated stance classification.
- **Chapter 5** focuses on models for automated rumour veracity classification and studies the effect of incorporating various sets of features across several models and datasets.
- **Chapter 6** presents a multitask learning approach to rumour veracity classification that allows us to leverage its connections with the tasks of rumour detection and stance classification.
- **Chapter 7** is concerned with making the decisions of a rumour verification model more transparent via an uncertainty estimation mechanism.
- **Chapter 8** concludes the thesis.

CHAPTER 2

Background

2.1 Prerequisites in Natural Language Processing

Machine Learning (ML) is field of study that gives "computers the ability to learn without being explicitly programmed" as defined by Samuel (1959). It studies algorithms and statistical models that extract patterns and make inference based on given data, in order to perform a specific task. **Natural Language Processing (NLP)** is a field at the intersection of Machine learning, Data Mining and Computational Linguistics, which studies the processing and analysing ('understanding') of natural human languages by a computer program. **Text Mining** is a sub-field of NLP focusing on knowledge discovery from text. **Deep Learning** is a sub-field of Machine Learning that studies a specific class of algorithms, artificial neural networks, that have several layers of connected 'neurons'. Throughout this thesis we mostly use methods from the class of deep learning and we will discuss the basic models in section 2.1.2. The relation and overlap between the above-mentioned fields of Machine Learning, Deep Learning, Natural Language Processing and Text Mining is shown in figure 2.1.

In this thesis we use Machine Learning methods to perform specific Natural Language Processing tasks related to the problem of online rumour spread. When the real world problem has been formulated as a Machine Learning task, a typical pipeline includes the steps of Data Collection, Data Preprocessing, Feature Engineering, Model Learning and Evaluation. This can then be repeated for several iterations based on the feedback from the evaluation step, tuning the model through parameter and feature selection at each iteration. This typical research process is demonstrated in Figure 2.2. Given the task, a suitable data source (e.g. News websites, Twitter, Instagram) and type (e.g. text, images) are identified and the dataset is collected. The data is then annotated by attaching task-specific labels to the data instances (e.g. sentences or articles). Next, the data is preprocessed, cleaned from

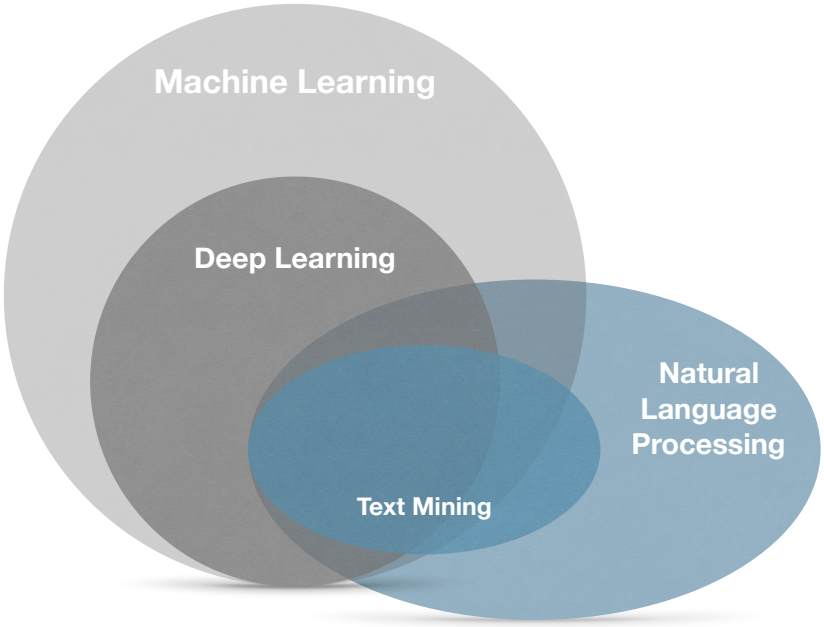


Figure 2.1: The overlap between the fields of Machine Learning and Natural Language Processing.

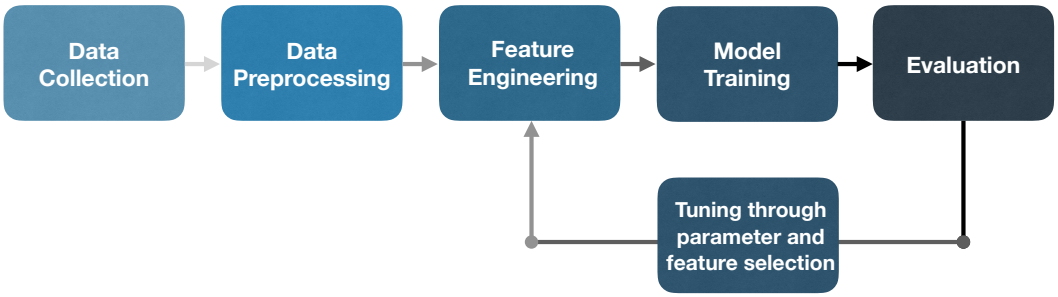


Figure 2.2: Typical Machine Learning pipeline.

noise and outliers, and converted into a vectorised representation suitable to be an input to the model. In the supervised learning setup the model learns a function $y = f(x)$ that maps an input x to an output y based on a set of training input-output pairs x, y. Finally, the effectiveness of the model is evaluated by comparing the output produced by the model $\hat{y}$ with true labels y on the testing data using suitable metrics. In the following sections we describe the basics of the algorithms and feature extraction methods used in this thesis, as well as evaluation strategies and metrics.

2.1.1 Text Representation

Textual data needs to be represented as numeric vectors to become a suitable input to a Machine Learning model. Word embeddings are techniques, which map words or phrases from a vocabulary to numeric vectors. In this section we describe two widely accepted approaches: Bag-of-Words (section 2.1.1.2) and word2vec (section 2.1.1.3), which have been used in this thesis, and briefly discuss further developments in representation learning. Prior to converting text into a vector representation it is often preprocessed, as discussed in the following section (2.1.1.1). Also, we briefly touch on the types of extra features that can be manually selected, depending on the task at hand, and added to the text representation in order to enrich it (section 2.1.1.4).

2.1.1.1 Preprocessing

Data preprocessing is an important step of the machine learning pipeline. This usually includes removing outliers, corrupted/implausible data points, and dealing with missing values in order to improve the dataset quality. Processing noisy, redundant and/or unreliable data may lead to misleading outcomes that are hard to interpret. Data preprocessing may include cleaning, instance selection, normalization and transformation.

Text preprocessing is a crucial step that is performed prior to converting documents into vector format and it very strongly affects the effectiveness of the resulting representation. The text preprocessing procedure differs depending on the task and data at hand, and frequently includes the following steps.

Lowercasing converts all letters in the document into lower case, such that all types of capitalisation, e.g. “Text”, “TExt” and “text” become the same entry for the model.

Stop words removal aims at removing from the text a number of very common words that are used as links in the sentence and typically convey little meaning, e.g. "a", "the", "at". Lists of stop words can vary, e.g. in this work we use one provided in Python NLTK package (Bird et al., 2009).

Removal of non alphanumeric characters removes punctuation and numbers from the document.

Stemming or lemmatisation are two steps that convert all of the words in the document into their shortened versions: either stems (base or root form) or lemmas (dictionary form that is based on its intended meaning). This step would turn "running" and "run" into the same entry.

Tokenisation splits the document into tokens. Most frequently these tokens are individual words however, depending on the purpose, bigrams (pairs of tokens) and n-grams (sequences of n tokens) can be used.

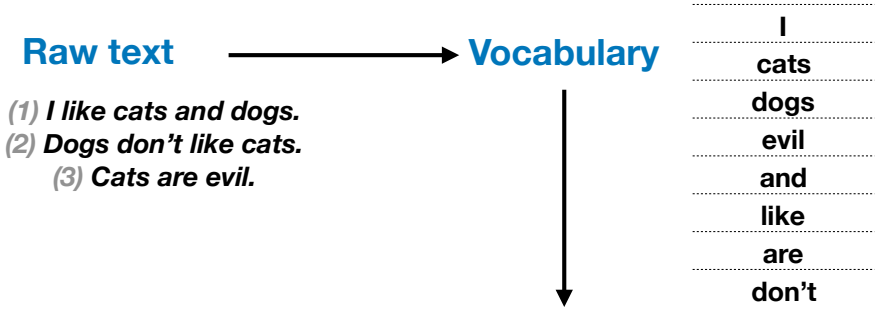
This list of preprocessing routines can be extended. In this work we are dealing with data collected from Twitter, which is often more noisy (less grammatical) than texts in books and newspapers, contains emoji and other special characters, therefore it is worth processing it with some additional steps.

Replacement of URLs, pictures and user mentions replaces URLs and/or user mentions with a chosen token so that they are processed in the same way by the model, e.g. "https://www.bbc.co.uk/" and "www.telegraph.co.uk" can be replaced by the token "URLURL".

Replacement of social media-specific symbols one may choose to replace frequently used emojis with their textual explanation.

2.1.1.2 Bag-of-Words representation

Bag-of-Words (BOW) is the simplest, basic way of representing words as vectors (Harris, 1954). It is illustrated in figure 2.3. A large corpus of raw text is tokenised into individual words and the vocabulary is built out of these unique entries. Then, given the vocabulary, each word is represented via one-hot-encoding, i.e. as a vector with dimensionality equal to the size of the vocabulary, with all entries zero, except for a one in the position corresponding to the position of the word. A sentence can then be represented as the sum of word vectors from which it consists. The



Bag-of-Words Representation

Vocabulary	Cats	Dogs	Sentence 1	Sentence 2	Sentence 3
I	0	0	1	0	0
cats	1	0	1	1	1
dogs	0	1	1	0	1
evil	0	0	0	1	0
and	0	0	1	0	0
like	0	0	1	0	1
are	0	0	0	1	0
don't	0	0	0	0	1

Figure 2.3: Bag-of-Words model.

Bag-of-Words model is a special case of the n-gram model, with $n = 1$, and hence can be generalised to any n by creating the vocabulary out of phrases of length n. The BOW approach is very common, however it has several drawbacks: (1) it produces sparse representations, where high dimensional vectors have very few non-zero elements, which leads to increased storage memory and computational time unless a special sparse representations are used[1]; (2) it ignores the word order in the sentence, and hence looses this important contextual information.

2.1.1.3 Distributed representation models

As opposed to sparse BOW representations, the models described in this section map words or phrases to vectors of real numbers in a low-dimensional space (relative to the vocabulary size), where each dimension is a latent feature. There are different ways of obtaining such mappings, often they are based on the distributional hypothesis: "a word is characterised by the company it keeps". Distributional

[1]In Python such implementations are available in the scipy.sparse package (Virtanen et al., 2019).

semantics studies methods for quantifying semantic similarities between linguistic items based on their distributional properties, because it is expected that linguistic items with similar distributions have similar meanings (Harris, 1954). This is in contrast to formal semantics that "seeks to understand linguistic meaning by constructing precise mathematical models of the principles that speakers use to define relations between expressions in a natural language and the world which supports meaningful discourse" (Aronoff and Rees-Miller, 2001). Models based on the distributional hypothesis require large text corpora to train on in order to obtain good word representations.

Mikolov et al. (2013a,b,c) proposed a neural language model called word2vec, which gained enormous popularity due to the ease of training, relatively short computation times and impressive semantic properties, leading to performance improvements in many tasks. This approach includes two models which are based on training a shallow (single hidden layer) feed-forward neural network (as described in section 2.1.2.1). The first model is called Continuous Bag-of-Words (CBOW) and it predicts a word given its context (figure 2.4, left). The other model, Skip-gram (SG), performs the inverse task of predicting the context, given a word (figure 2.4, right). The weights matrix of the model is then used to extract word vectors. These models were trained on the Google News dataset, a large corpus of news articles, containing about 100 billion words. Inputs are generated using a sliding window (see 2.4) over the news articles and each word is represented using one-hot encoding. The dimensionality of the resulting vector is a parameter that can be tuned depending on the amount of available data. The quality of the word embeddings was tested using a set of five types of semantic questions and nine types of syntactic questions that could be resolved using vector arithmetic and cosine distance measures. Word vectors trained by these models encode a lot of linguistic regularities, like gender, verb tense, country-capital relations and more. These regularities reveal themselves through vector arithmetic: vector('king') - vector('man') + vector('woman') = vector('queen').

Further works investigated the workings of those models and proposed additional improvements. Levy and Golberg explored the reasons behind the success of the word2vec model in (Levy and Goldberg, 2014) and show that it is implicitly factorises a word-context matrix, whose cells are the pointwise mutual information (PMI) of the respective word and context pairs, shifted by a global constant. Baroni et al. (2014) performed extensive evaluation of context-predicting models such as word2vec and context-counting models such as BOW on a wide range of lexical semantics tasks and across many parameter settings. They show that the context-

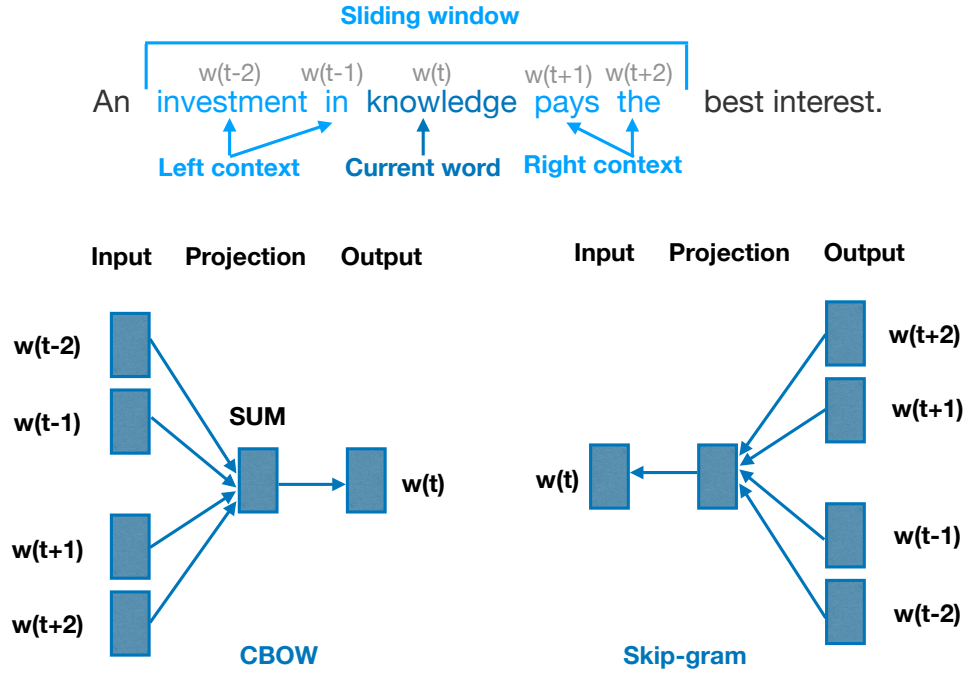


Figure 2.4: Word2vec models. Left: Continuous Bag-of-Words (CBOW); Right: Skip-gram.

predicting models beat their count-based counterparts. Later Levy et al. (2015) show that much of the performance gains of word embedding models should be attributed to certain system design choices and hyper-parameter optimisations.

Pennington et al. proposed another context-based word embedding model Glove (Pennington et al., 2014), a log-bilinear regression model, which uses not only local context window but also global context information. The Glove model showed comparable performance to word2vec and also gained a lot of popularity.

There are models that propose sentence and document level embeddings into a vector space based on the principle of compositionality, that the meaning of a complex expression is determined by the meanings of the expressions it is composed of, and the rules used to combine them: Palangi et al. (2016); doc2vec by Le and Mikolov (2014); Blunsom et al. (2014).

Out of the recent advances in representation learning the most prominent models are ELMO (Peters et al., 2018), OpenAI GPT (Radford et al., 2018) and BERT (Devlin et al., 2018) that are providing context-dependent word representations.

ELMO are deep contextualized word representations that model the complex characteristics of word use, such as syntax and semantics, and how these uses vary across linguistic contexts, i.e. polysemy. ELMO word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pretrained on a large text corpus.

OpenAI GPT, Generative Pre-trained Transformer, creates effective word representations by generative pre-training of a language model on a diverse corpus of unlabeled text, followed by discriminative fine-tuning on each specific task.

BERT, which stands for Bidirectional Encoder Representations from Transformers, is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. BERT was shown to obtain state-of-the-art results on eleven NLP tasks.

These contextualised representation models are more complex than word2vec described above and require not only large amount of data but also significant computational resources to train from scratch. However, as pre-trained models are often publicly available they are gaining popularity among NLP practitioners. As the above-mentioned representations are relatively recent advances, in this work we mainly used word2vec for our word representations, however it is possible to extend this to novel contextual embeddings in future work.

2.1.1.4 Task-specific extra features

Often document representation is concatenated with other extra features that are either also extracted from the text or contained in meta-information. These features are always task specific and added to improve models performance. Some of the common extra features are summarised below.

Part-of-Speech (POS) tags , have been shown to be useful for many NLP tasks, in particular for Named Entity Recognition. They can be represented as a vector where each dimension corresponds to one of the possible parts of speech and only one of the dimensions will have value of 1 and zero elsewhere.

Punctuation that was removed during the preprocessing stage can be re-introduced as either embeddings or binary features indicating, for example, presence of exclamation mark or question mark. This proves useful for tasks of stance classification and sentiment analysis.

Lexicon-based features that indicate the presence of certain words that might be associated with the target variable. There are many available lexicons for different topics, such as lists of negation words, swear words, and words with positive or negative sentiment. One of the most comprehensive lexicons that includes multiple topics is Linguistic Inquiry and Word Count (LIWC) (Pennebaker et al., 2001).

These extra features can be platform-specific and reflect meta-information accompanying social media posts.

User information features include the profile description written by a user, profile age, whether the profile is verified by the platform and a collection of previous posts.

Interactions with other users includes the number of 'likes' and 'shares' of an individual post, number of followers and followees of the user, or the user's full connection network.

Presence of post attachments can be used as a binary feature indicating whether a URL and/or pictures are attached to the post.

Location information can be provided as part of user information or each individual post can be geotagged and can be used as a feature.

The process of feature selection is performed in order to determine whether a feature is relevant to the task. Feature selection may follow different strategies depending on the amount of data and computational resources available. It can be simply based on indicators of feature correlation with the target variable without considering the relationships between the features. Frequently the usefulness of a feature is tested by training a model with various subsets of supposedly relevant features and evaluating its performance on the development set. Feature selection helps increase model interpretability, reduce computation time and prevent overfitting (Friedman et al., 2001).

2.1.2 Algorithms

In this section we describe the main algorithms that are widely used in our experiments and in the field of Natural Language Processing in general. We only consider classification algorithms in this section as this thesis mainly used classification approaches. More concrete details on our experiments will be presented in the corresponding chapters.

Some of the most common classifiers are Logistic Regression (LR), Support Vector Machines (SVM) and Random Forest (RF). They are even sometimes referred to as "off-the-shelf" classifiers as they are relatively easy to tune and fast to train. However, recently deep learning models have gained popularity and achieved state-of-the-art in many NLP tasks, such as Recognising Textual Entailment, Question Answering, Sentiment Analysis and more (Wang et al., 2019), often reaching human performance. Thus often leaving the baseline role to the above-mentioned LR, SVM and RF. We will explain in more detail the work of several types of deep learning algorithms, namely Feed Forward and Recurrent Neural Networks, as these were at the basis of our experiments and only briefly describe Logistic Regression, Support Vector Machines and Random Forests, which were used as baselines (for more information on LR, SVM and RF we recommend the reader to look into the materials referred below).

Logistic regression (LR) (Cramer, 2002) is a statistical model that uses a logistic function to model a binary dependent variable. It models the probability of a certain output based on the input. In order for it to be used as a classification model a cutoff value (often 0.5) is introduced, such that inputs with probability greater than the cutoff belong to one class, below the cutoff to the other (Friedman

et al., 2001). Logistic regression can be viewed as a special case of a Neural Network. Illustration on figure 2.5 is also a representation of LR model if activation is sigmoid.

Support vector machine (SVM) (Cortes and Vapnik, 1995) is a non-probabilistic supervised learning model that finds a $(n-1)$-dimensional hyperplane that separates training instances in n-dimensional space with the largest margin between classes. Testing instances are then mapped into the same space and the class is determined depending on which side of the separating plane they lie. SVMs can also efficiently perform a non-linear classification using the kernel trick, by mapping their inputs into high-dimensional feature space and then fitting the maximum-margin hyperplane in that transformed feature space (Boser et al., 1992).

Random Forest (RF) (Breiman, 2001) is an ensemble learning method that works by constructing a number of 'weaker' classifiers called decision trees. In decision trees an instance goes through a number of forking branches that represent feature values that lead to class labels, then class is determined at the leaf node. Random Forest uses the mode of the classes predicted by several decision trees, thus producing a more robust classifier that is less prone to overfitting the training data than an individual decision tree.

2.1.2.1 Feed Forward Neural Networks

Artificial Neural Networks are inspired by biological neural networks in the human brain. They are represented as layers of connected neurons, each performing a weighted summation, followed by a non-linear activation function (Friedman et al., 2001). The computation performed by a single neuron is illustrated in figure 2.5. It is equivalent to LR model if activation function is sigmoid.

Figure 2.6 shows the simplest neural network architecture with one hidden layer. In this network, each neuron of the previous layer is connected to each neuron of the following layer, this type of connection is called a dense layer. This network is an example of a feed forward network, in which connections between the nodes do not form a cycle. Training of a neural network consists of forward propagation and backpropagation. Given an input $X \in \mathbb{R}^n$, ground truth labels y and a set of randomly initialised weights $W \in \mathbb{R}^{n \times m}$, where m is the number of neurons per layer. Forward propagation computes the output $\hat{Y}$. For the architecture in figure 2.6 it will be of the following form (in matrix notation), as shown in equations 2.1–2.3.

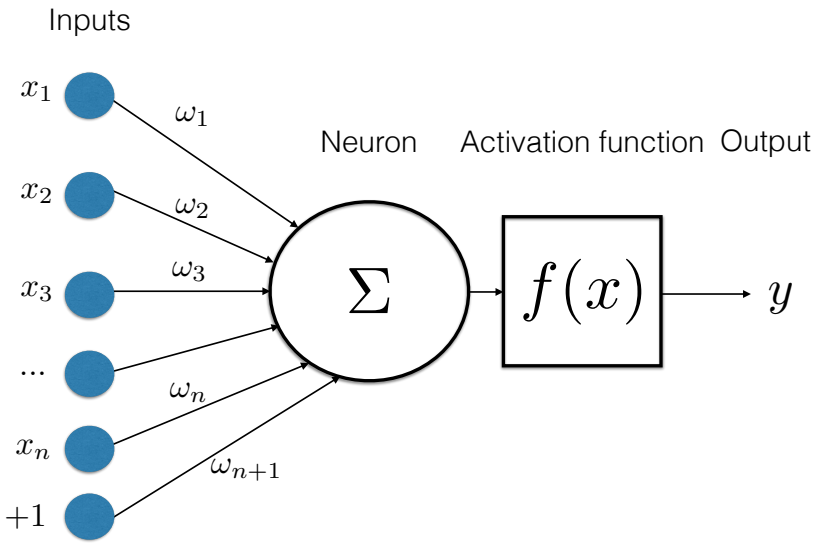


Figure 2.5: Single neuron computation.

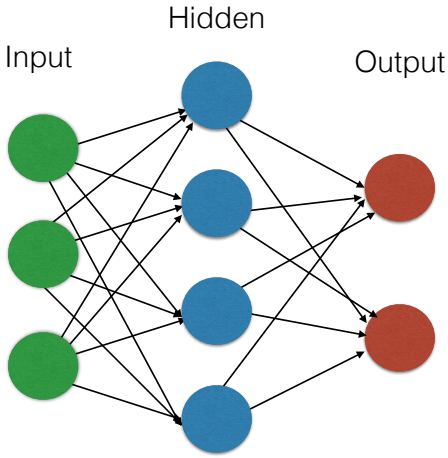


Figure 2.6: Feed forward connection of neurons: each neuron of previous layer connected to each neuron of the following layer without loops.

$$H = f(W_x \cdot X), \tag{2.1}$$

$$\hat{Y} = g(W_y \cdot H), \tag{2.2}$$

$$E = L(Y, \hat{Y}), \tag{2.3}$$

where H is the output of the hidden layer, W_x and W_y are weights for hidden and output layers, f and g are non-linear activation functions, $\hat{y}$ is the predicted output, y the target output, L the loss function and E the loss. A common choice for activation functions are hyperbolic tangent (equation 2.4), sigmoid (equation 2.5) and Rectified Linear Unit (ReLU) (equation 2.6).

$$f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{2.4}$$

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}, \tag{2.5}$$

$$f(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}. \tag{2.6}$$

In order for the network to improve, it needs to update weights during back-propagation using Stochastic Gradient Descent (SGD) (equation 2.7).

$$W = W - \epsilon \frac{\partial E}{\partial W}, \quad \forall W \in [W_y, W_x], \tag{2.7}$$

where $\epsilon > 0$ is the learning rate and the chain rule is used to find partial derivatives. Updates of the weight matrices can be performed either after each new input instance (stochastic), a mini-batch of instances or a pass through the whole training data.

We have described the main idea, and the simplest version, of optimising the parameters of a neural network via gradient descent. There exist many algorithms to optimise the weights based on gradient descent that often achieve convergence faster, and hence are frequently used for neural network optimisation (Ruder, 2016), such as Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012) and Adam (Kingma and Ba, 2014).

This simple example illustrates the basic principles of how neural models work, however more complex architectures can be developed by using different combinations of nodes.

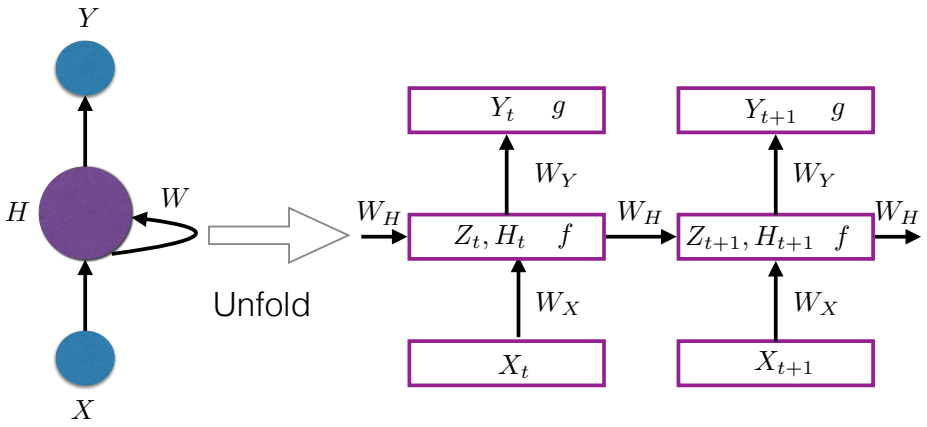


Figure 2.7: Recurrent Neural Network illustration.

2.1.2.2 Recurrent Neural Networks

In this section we describe Recurrent Neural Networks (RNNs), a class of artificial neural networks where connections between units form a directed cycle. They are well-suited for sequential and time-series data due to their internal memory property. Hence, they are often applied to Natural Language Processing tasks, for example lexical entailment (Bowman et al., 2015), sentence embedding (Palangi et al., 2016) and sequence generation (Graves, 2013).

Recurrent neural networks operate on the linear progression of time, combining the previous time step and a hidden representation into the representation for the current time step. Figure 2.7 illustrates the work of a recurrent unit and how it can be 'unrolled in time'. It contains a hidden state, depicted as H_t in figure 2.7, that is updated using not only new information at time step $t+1$, but also past information via the hidden state of the previous time step, hence it has a memory property. Forward propagation through the simplest version of a recurrent network is shown in equations 2.8–2.12 (Elman, 1990).

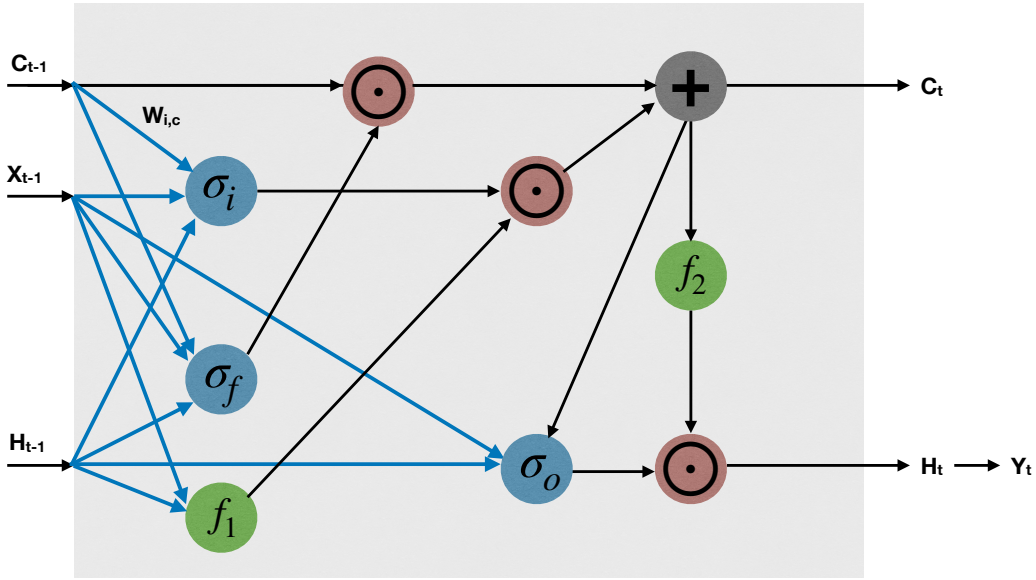


Figure 2.8: Long Short Term Memory Unit Flow. Blue lines mean weighted summation with individual weights per-pair of instance, black lines mean passing the instance.

$$Z_t = W_X \cdot X_t + W_H \cdot H_{t-1}, \tag{2.8}$$

$$H_t = f(Z_t), \tag{2.9}$$

$$\hat{y}_t = g(W_Y \cdot H_t), \tag{2.10}$$

$$E_t = L(Y_t, \hat{Y}_t), \tag{2.11}$$

$$E = \sum_{t=1}^{T} E_t, \tag{2.12}$$

where x_t is the input at time step t, $\hat{y}_t$ the output, y_t is the true label, W the sets of weights, f and g are non-linear activation functions, L is the loss function, E_t the loss at time t and E the overall loss. H_{t-1}, the hidden state at time $t-1$, is used in the calculation of the hidden state H_t (equations 2.8 and 2.9) and H_0 is initialised randomly. Recurrent Neural Networks are trained using 'backpropagation through time', which is analogous to standard backpropagation, where the errors and gradients at each time step for one training example are summed up.

There are different types of RNN units, one of which is Long Short-Term Memory (LSTM) blocks (Gers et al., 2000), which include input, output and forget gates within the unit (see illustration in figure 2.8). RNNs often suffer from the problem of vanishing gradients, i.e. gradients with very small values that prevent the weights from changing values, which can completely stop training in the worst case. This problem is often caused in the process of backpropagation that uses chain rule, which means multiplying a lot of gradient values together, and when gradients are in the range $(0, 1)$ that leads to vanishingly small values. LSTMs overcome the vanishing gradient issue of RNNs and allow for the capture of long-term dependencies in the data (GERS, 2001; Graves, 2012). Forward propagation of a LSTM unit is shown in equations 2.13–2.17.

$$C_t = f_t \odot C_{t-1} + i_t \odot f_1(W_x \cdot X_t + W_H \cdot H_{t-1}), \tag{2.13}$$

$$H_t = o_t \odot f_2(C_t), \tag{2.14}$$

$$i_t = \sigma(W_{i,X} \cdot X_t + W_{i,C} \cdot C_{t-1} + W_{i,H} \cdot H_{t-1}), \tag{2.15}$$

$$o_t = \sigma(W_{o,X} \cdot X_t + W_{o,C} \cdot C_t + W_{o,H} \cdot H_{t-1}), \tag{2.16}$$

$$f_t = \sigma(W_{f,X} \cdot X_t + W_{f,C} \cdot C_{t-1} + W_{f,H} \cdot H_{t-1}), \tag{2.17}$$

where X_t is the input, $\cdot$ denotes matrix multiplication, $\odot$ element-wise mul-

tiplication, t is the current time step, C_t the cell state at time t, H_t the hidden state at time t (H_0, C_0 are initialised randomly), i_t, o_t, f_t are input, output, forget gates, σ is a sigmoid function and f_1, f_2 are non-linear activation functions.

Training a neural network is a hard problem as it involves an optimisation over a high-dimensional parameter space to find the set of parameter (weights) that minimise predictive error, which is non-convex and contains local minima.

Neural networks also require choosing multiple hyper-parameters that may strongly affect the model performance, such as number of hidden layers, number of neurons per layer, non-linear activation function per layer, learning rate, regularisation technique and its parameters, e.g. dropout rate, and of course, initial weight initialisation. Several works (Greff et al., 2016; Jozefowicz et al., 2015) have explored the effects of different hyper-parameters on RNNs by altering them and applying LSTM RNN to a set of problems, thereby producing general recommendations for training LSTM networks. Greff et al. (2016) suggested that learning rate and hidden layer size are the most important parameters, and that the forget gate and output activation function are the most crucial components. Also input and forget gate coupling ($f_t = 1 - i_t$), and removing peep-hole connections (peep-hole connections refer to adding C_t and C_{t-1} variables into the gate equations 2.15–2.17, allowing gate layers look at the cell state), do not hurt the performance while reducing computational complexity. Jozefowicz et al. (2015) highlighted that the use of dropout is very important and also recommended to add a bias of 1 to the forget gate of the LSTM. Improvements in model performance often come from supplying a sequence to the network in both directions: forward and backward, therefore creating a bi-directional RNN (Schuster and Paliwal, 1997).

2.1.2.3 Recursive Neural Networks

Recursive neural networks are an extension of recurrent neural networks that are good at capturing tree-like structures in the data. Recursive Neural Networks have been successful in learning sequence and tree structures in natural language processing tasks such as semantic parsing (Socher et al., 2013, 2011). Recursive neural networks operate on any hierarchical structure, combining child representations into parent representations. Recurrent neural networks are in fact recursive neural networks with a particular structure: that of a linear chain. The gradient is computed using Backpropagation Through Structure (BPTS), which is principally the same as general backpropagation. There are two differences resulting from the tree structure: derivatives are split at each node, and derivatives of W from all nodes are summed.

Figure 2.9: Model validation approaches.

Neural networks are very flexible architectures with a large number of parameters (weights) to optimise. That makes them require large amounts of data to achieve the best results. Due to their flexibility they are also prone to fitting training data very precisely, i.e. overfitting, potentially including the noise contained in the data. That could lead to problems with generalisability, therefore there are multiple methods that help prevent this situation including weight regularisation (van Laarhoven, 2017), dropout (Srivastava et al., 2014) and batch normalisation (Ioffe and Szegedy, 2015).

2.1.3 Evaluation

Given a model, we need to evaluate its performance and ability to generalise to unseen data. In this section we describe two model validation approaches and predictive performance metrics.

2.1.3.1 Validation approaches

In order to estimate how well a predictive model will perform in practice, we need to test it on a set of data that was not used for training the model. Models that fit the training data very precisely, may also fit the noise inherent in real-world datasets, and hence are not likely to perform well on the new, previously unseen

inputs. This situation is called overfitting. Here we describe two commonly adopted model validation approaches that test for generalisability and overfitting: splitting the dataset into training, development and testing subsets, and cross-validation (Friedman et al., 2001).

Training, development and testing split means dividing the dataset into three separate subsets as shown in figure 2.9 (top). The training set is used for training the model; development for intermediate assessment of the model, while tuning hyper-parameters; and testing set only for a final evaluation.

Cross-validation is a training and evaluation scheme in which the dataset is partitioned into k folds. Then, k iterations of training and testing are performed, switching the testing fold at each iteration as shown in figure 2.9 (bottom). The final performance is an average of the model's performance over the folds. The k value can be chosen arbitrarily depending on the data at hand, e.g. it can be equal to the number of instances in the dataset, known as leave-one-out cross-validation. The split into folds is often randomised, however it can also be assigned using prior information, e.g. stratified to balance parameters of choice in training and testing sets. One could also perform validation by combining the two above-mentioned approaches, by splitting the dataset into training/development/testing subsets and then performing n-fold cross-validation on the training set. In this work we often use leave-one-event-out cross-validation, where each fold corresponds to an independent real-world event, as this more accurately reflects the real world scenario.

2.1.3.2 Evaluation metrics

In this section we describe a set of common metrics used to evaluate the performance of predictive models. They work by comparing the set of model predictions against the true labels (Friedman et al., 2001).

Confusion matrix is a matrix C in which elements C_{ij} are equal to the number of observations with true class label i but predicted to be in class j. The confusion matrix for binary classification is presented in table 2.1, where: True Positives (TP) are the correctly predicted instances of the True class; False Positives (FP) are instances from the False class predicted as True; False Negatives (FN) are instances from True class predicted as False; True Negatives (TN) are the correctly predicted instances of the False class.

		Actual class	
		True	False
Predicted class	True	TP	FP
	False	FN	TN

Table 2.1: Confusion matrix for binary classification. TP: True Positives, FP: False Positives, FN: False Negatives, TN: True Negatives.

Accuracy is defined as the fraction of correctly classified instances. Equation 2.18 shows an expression for the accuracy of the binary classification task.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}. \tag{2.18}$$

The following metrics (Precision, Recall and F-score) are defined for the case of the binary classification.

Precision is defined as the number of correctly identified positive instances divided by the number of predicted as positive, as shown in equation 2.19.

$$P = \frac{TP}{TP + FP}. \tag{2.19}$$

Recall is defined as the number of correctly identified positive instances divided by the number of instances in the positive class, as shown in equation 2.20. Recall characterises the ability of the model to find all of the positive samples.

$$R = \frac{TP}{TP + FN}. \tag{2.20}$$

F1 score is the harmonic average of the precision and recall. The formula is shown in equation 2.21.

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} = \frac{2TP}{2TP + FP + FN}. \tag{2.21}$$

In the multi-class case, Precision, Recall and F score can be calculated as an average of per-class metrics. Two types of averaging can be performed: micro- or macro-. Micro-averaging means aggregating the contributions of all classified instances to compute the average metric i.e. averaging the individual True Positives, False Positives, and False Negatives of the system for different classes. Macro-averaging means calculating metrics for each label, and then using their unweighted

mean. Accuracy and micro-averaged F scores are widely used for evaluation of supervised learning tasks giving equal weights to each classification instance. Macro-averaged F scores are preferred when the dataset contains a significant class imbalance, and high performance is desirable for all classes, as macro-averaged F score gives equal weight to each of the classes.

RMSE is a quadratic scoring rule that measures the average magnitude of the error. It is often used in evaluation of regression tasks. In chapter 5, the RMSE score is used to evaluate how the levels of model certainty relate to the correctness of its predictions.

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y_j})^2}, \tag{2.22}$$

where y_j is the true value and $\hat{y_j}$ is the predicted value.

Even though the above-mentioned metrics are standard for evaluation of machine learning models, there are a number of ML tasks representing real world problems where they are not suitable. For example, when the dataset is imbalanced, a model that always predicts the majority class will get a high accuracy score, which would be a misleading representation of its performance. In such cases, using the macro-averaged F1-score is more appropriate as it weights performance on each of the classes equally. However, depending on the problem, not all of the classes are equally important to detect. In some tasks like summarisation, machine translation and natural language generation, the output is not strictly defined as the 'correct human answer' can vary. Custom evaluation metrics can be defined to better fit the task at hand, for example in the rumour verification shared task alternative metrics were defined (see section 5.5).

2.2 Related work

In this section we present definitions of the rumour detection, verification and rumour stance classification tasks and show how they fit with other Natural Language Processing tasks. We present the rumour resolution process as a pipeline of several subtasks and provide an overview of the works relevant to the field of automated rumour verification.

2.2.1 Rumour verification pipeline: tasks and definitions

Definitions of rumour

There is no unique definition of a rumour in recent publications on automatic rumour detection and verification. Some works define rumour as a false statement (Qazvinian et al., 2011; Liang et al., 2015). Qazvinian et al. (2011) describes a rumour as a "statement whose truth-value is unverifiable or deliberately false", thus also including statements that can not be verified. Liang et al. (2015) refers to rumours as "information whose truth and source are unreliable, and are likely to be generated under emergency situation, causing public panic, disrupting the social order", focusing on the reliability of the source, i.e. the users spreading the rumour.

Allport and Postman (1965) in their work "The psychology of rumor", that is considered a milestone of social psychology, define rumour as "a specific (or topical) proposition for belief, passed along from person to person, usually by word of mouth, without secure standards of evidence being present." This definition treats rumour as an unverified statement that can be eventually resolved as true or false.

Thus the majority of works performing detection and verification (Zubiaga et al., 2018c) use the definition supported by studies in social psychology, where a rumour is a piece of circulating information whose veracity status is yet to be verified at the time of posting. This aligns with definitions in Peterson and Gist (1951) and DiFonzo and Bordia (2007) of a rumour as "an unverified account or explanation of events circulating from person to person and pertaining to an object, event, or issue in public concern". In this thesis we use the following definition that is written based on the definition above:

> Rumour is a statement related to an event and unverified at the time of posting and circulation, rumour is (1) verifiable as opposed to reflecting an opinion, comment or emotion and (2) spreads widely, i.e. is impactful, hence its resolution is important, and the claim is 'check-worthy'.

Often the term 'fake news' is used in relation to rumours or false rumours, however in this work we prefer to make a distinction between rumours and fake news, using the term 'fake news' for news articles published by news outlets that are intentionally and verifiably false (Shu et al., 2017).

Rumour verification task under different definitions of rumour

Since a rumour as a concept has several interpretations, these affect how the rumour detection and verification tasks are defined in the literature.

Where a rumour is defined as a false statement (Qazvinian et al., 2011; Cai et al., 2014; Liang et al., 2015) the process of identifying posts with false claims is referred to as rumour detection. Liang et al. (2015) also refers to it as rumour identification.

As we follow the definition of a rumour being a claim with questionable veracity (Zubiaga et al., 2018c) as opposed to false statement, we define the rumour detection task as the classification of a given claim as either a *rumour* or *non-rumour* (opinion, chat), thus deciding whether a claim is check-worthy. It follows that rumour verification is the process of determining whether a given rumour is *true*, *false* or *unverified*. These tasks can be viewed as part of the rumour resolution process.

Ma et al. (2016, 2017, 2018a) follow the same definition of rumour, however they collate the tasks of rumour detection (binary classification) and verification (3-way classification) into a 4-way classification task of categorising claims as either *true*, *false*, *unverified* or *non-rumour*. They refer to this task as rumour detection.

Li et al. (2019a) acknowledge the existence of a variety of definitions, and they perform the task of rumour verification in accordance with Zubiaga et al. (2018c), however they choose to refer to it as either rumour detection or verification interchangeably.

These differences in rumour and task definitions are often reflected in the approaches to dataset creation, and hence, one should be careful when using publicly available datasets. Researchers need to understand the definitions employed by the authors of the dataset as well as the data collection and annotation process in order to use them in conjunction with suitable tasks. Often even if datasets are of similar structure and collected from the same platform it may not be possible to join them in a unified larger dataset. Chapter 3 provides a detailed description of the datasets chosen for analysis in this thesis.

Rumour resolution pipeline

Rumour resolution is a process that can be split into a number of sub-tasks. In Zubiaga et al. (2018c) it is defined as a pipeline involving four steps, as is shown in figure 2.10: (1) rumour detection, determining whether a claim is worth verifying rather than the expression of an opinion; (2) rumour tracking, collecting sources

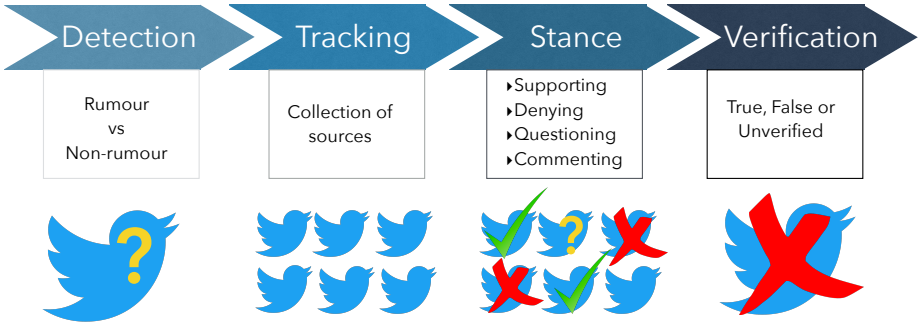


Figure 2.10: Rumour resolution pipeline.

and opinions on a rumour as it unfolds; (3) stance classification, determining the attitude of the sources or users towards the truthfulness of the rumour, and (4) rumour verification, as the ultimate step where the veracity value of the rumour is predicted.

While this is very common and general view of rumour resolution, naturally, depending on the specific use case, the pipeline can be adjusted: components can be modified, added or removed. For example, components can be split into further sub-tasks, and become more or less fine-grained classification tasks with respect to the number of different classes. For instance, veracity classification can be split into analysing social media posts, assessing the provenance of accompanying images or other media content and gathering relevant news articles.

These steps can be performed at different times in the life-cycle of a rumour, making this a time-sensitive process. Ideally, rumours can be resolved as either true or false. However, they can also remain unverified when there is insufficient evidence to determine their veracity (Caplow, 1947).

In the following sections we discuss the tasks of rumour detection (section 2.2.3), verification (section 2.2.5) and stance classification (section 2.2.4) in more detail. We omit rumour tracking and consider it outside of the scope of this thesis.

Relations between rumour verification and credibility assessment tasks

The task of rumour verification is closely related to the tasks of fact checking and claim credibility assessment.

One of the first works tackling the problems of rumours on social media were works on rumour credibility classification (Castillo et al., 2011; Mendoza et al., 2013; Gupta and Kumaraguru, 2012). Fogg and Tseng (1999) describe credibility

as 'believe-ability', a perceived quality, which means it does not reside in an object, a person, or a piece of information. Credibility perceptions result from evaluating multiple dimensions simultaneously, the key ones being trustworthiness and expertise. Trustworthiness captures the perceived goodness or morality of the source, while the expertise dimension captures the perceived knowledge and skill of the source. On the contrary, rumour veracity is an actual quality of an object, a claim. Castillo et al. (2011) defined a credible rumour as a news claim offering reasonable grounds for being believed. Thus claim credibility classification is the task of identifying whether a claim seems probable/truthful to users. Even though the estimated credibility of the rumour does not reflect its actual veracity, studies of claim credibility assessment were one of the first ones in the field and since the tasks of credibility and veracity classification are related, they identified useful features that also can be exploited for veracity classification. Castillo et al. (2011) analysed microblog postings related to "trending" topics, and classified them as credible or not credible, based on features extracted from them.

Gupta and Kumaraguru (2012) analysed the credibility of information in tweets corresponding to fourteen high impact news events, performing credibility ranking rather than binary classification. They identified the important content- and source-based features, which can predict the credibility of information in a tweet. Prominent content-based features were the number of unique characters, swear words, pronouns, and emoticons in a tweet, and user based features like the number of followers and length of username. O'Donovan et al. (2012) shows that URLs, mentions, retweets and tweet length are strong credibility cues.

A separate line of work involves analysing the credibility of sources. When source assessment is performed, information coming from a trustworthy source is also considered credible. Canini et al. (2011) analysed the credibility of sources of information rather than individual claims, and observed that content and network structure act as prominent features for user credibility ranking.

Relations between rumour verification and fact-checking tasks

The concepts of fact checking and verification are studied by journalists as well as researchers, often giving different definitions to both concepts, where some are more broad and some are narrowed down. While both communities acknowledge those concepts as separate, fact-checking and verification are so tightly related that they are often considered as either overlapping, complementary or one being a part of another, as each is about confirming or debunking information (Thorne and Vlachos, 2018).

Silverman (2013) defines verification as a "discipline that lies at the heart of journalism, and that is increasingly being practised and applied by other professions" and fact-checking as a "specific application of verification in the world of journalism". Silverman considers verification "a fundamental practice that enables fact checking"[2].

While Mantzarlis[3] defines verification as "a process that evaluates the veracity of a story before it becomes the news" and fact-checking as "a process that occurs post publication and compares an explicit claim made publicly against trusted sources of facts".

First Draft[4] produces a similar definition of verification as "the process of determining the authenticity of information posted by unofficial sources online, particularly visual media" and fact-checking as "the process of determining the truthfulness and accuracy of official, published information such as politicians statements and news reports"[5]. Thus the process of verification focuses on "source, date, location" and concentrates on the reliability of the origin of a claim, while fact-checking addresses the claims logic, coherence and context. Figure 2.11 shows the difference between fact-checking and verification according to Mantzarlis.

Vlachos and Riedel (2014), working on automating the process of fact-checking, define fact-checking as the assignment of a truth value to a claim made by public figures, such as politicians or pundits, in a particular context. They point out that fact-checking can be viewed as a classification task with different levels of granularity: simply binary with *true* and *false* labels, or more complex, taking into account many aspects leading to claims not being strictly true or false, such as time, speaker, multiple sources and interpretations. Vlachos and Riedel (2014) choose to restrict the task to claims that can be fact-checked objectively, which is not necessarily the case for statements assessed by journalists.

Given the broad context of term usage for verification and fact-checking, we focus on making the distinction between the two with respect to the work we perform in this thesis and its limitations. We are concerned with automating the process of rumour verification, where rumours are claims made by Twitter users, often emerging during crisis events, that are characterised by a lack of precise detailed official information at the early stages. The importance of attempting verification

[2] `https://verificationhandbook.com/additionalmaterial/verification-and-fact-checking.php`

[3] `https://www.poynter.org/fact-checking/2015/will-verification-kill-fact-checking/`

[4] First Draft is an organisation dedicated to supporting journalists, academics and technologists working to address challenges relating to trust and truth in the digital age.

[5] `https://medium.com/1st-draft/information-disorder-part-1-the-essential-glossary-19953c544fe3`

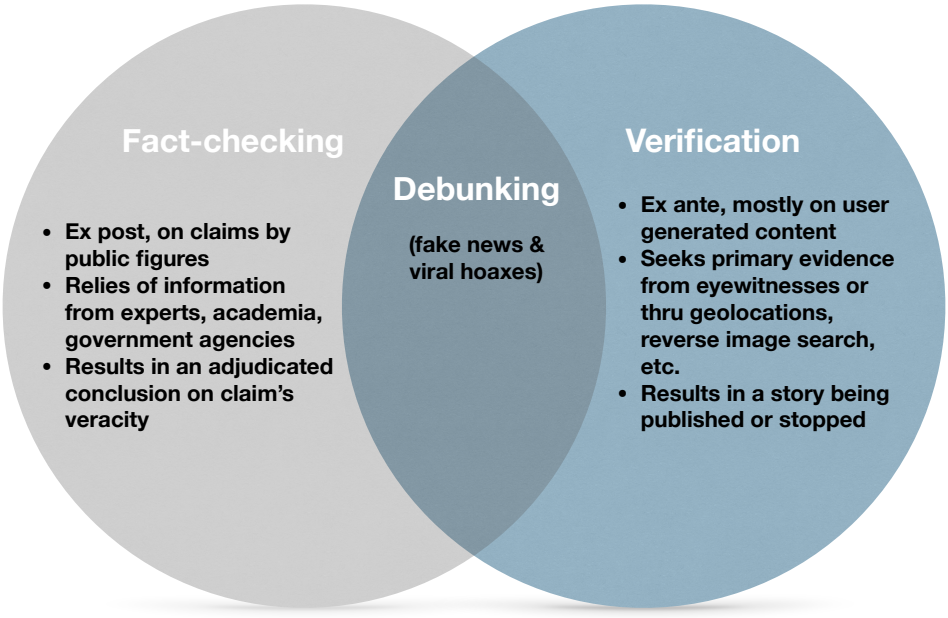


Figure 2.11: The difference between fact-checking and verification, as defined by Mantzarlis.

as early as possible, to have better chances of control over the rumour spread, makes it a time-sensitive task. By contrast, in fact-checking the sources of evidence are considered given or existing, e.g. Wikipedia or Freebase. This has led to many approaches incorporating knowledge bases as input for automated fact-checking systems (Nakashole and Mitchell, 2014; Ciampaglia et al., 2015; Thorne and Vlachos, 2018).

Working with cases where rich, credible textual sources or structured knowledge bases are likely to be unavailable or sparse, we therefore consider the following aspects of the verification task: (1) a comparison between assertions made in a candidate text and external world knowledge, as well as the entailment relation between them; (2) the assessment of the macro level user behaviours, their interactions and distribution of content, to predict whether the claims in the content are true or false (Derczynski et al., 2017). In this thesis we mostly focus on the second aspect concerning user behaviour and consider incorporation of external knowledge a part of future work.

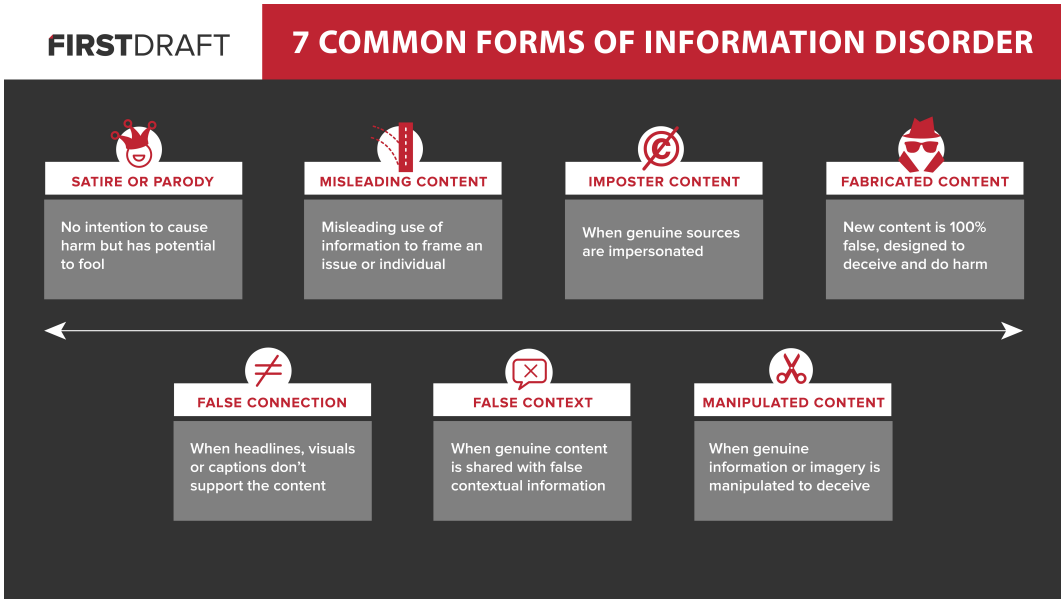


Figure 2.12: Seven types of misinformation defined by the First Draft organisation. (Image credit: Claire Wardle, 2017, `https://medium.com/1st-draft/information-disorder-part-3-useful-graphics-2446c7dbb485`.)

Types of rumours and misinformation

False information that is spread online is very heterogeneous and can be split into different types, according to its properties.

One of the early rumour taxonomies was introduced by Knapp (1944), who suggested to divide rumours into three classes according to their effect on the reader: (1) "pipe-dream" rumours, i.e. rumours that lead to wishful thinking; (2) "bogie" rumours, i.e. those that increase anxiety or fear; and (3) "wedge-driving" rumours, i.e. those that generate hatred.

Also, false information can be classified by the intent behind its spread. Misinformation means false or inaccurate information. This term is also often used to refer to false information that is unintentional, such as genuine mistakes in, for example, inaccurate photo captions, dates, statistics, translations or when satire is taken seriously. It is opposite in intent to disinformation: fabricated or deliberately manipulated audio/visual content with the intention to deceive[6]. In this work, rumour verification does not aim to determine intent, thus does not distinguish between misinformation and disinformation.

The First Draft organisation suggests distinguishing seven types of online

[6] `https://medium.com/1st-draft/information-disorder-part-3-useful-graphics-2446c7dbb485`

Figure 2.13: Ten types of misleading content as defined by the EAVI organisation. (Image credit: EAVI, 2017, `https://eavi.eu/beyond-fake-news-10-types-misleading-info/`.)

false content, as shown in figure 2.12, where the term misinformation is used in its broad meaning of false information. While the European Association for Viewers Interests (EAVI) Media Literacy organisation provides their own classification of misinformation categorized into ten types, as shown in figure 2.13. The types of rumours we work with in this thesis are mostly misinformation that arises in the context of ambiguity, when the meaning of a situation is not readily apparent, or potential threat, when people feel an acute need for security (DiFonzo et al., 1994). Those rumours are often from the false connection and false context categories (as defined in figure 2.12), however a more thorough evaluation of the dataset is needed to provide strict categorisation of these rumours.

Zubiaga et al. (2018b) propose distinguishing two types of rumours according to their timeline: (1) long-standing that are discussed for long periods of time, and (2) new rumours that emerge during breaking news and generally are resolved quickly.

In this work we concentrate on the second type, in which the challenging aspect is the fact that updates associated with breaking news stories are often released piecemeal, hence there is less information available, but also more important as early verification is crucial in crisis situations.

2.2.2 Journalistic approach to rumour verification

Often information about newsworthy events initially appears on social media, posted by witnesses, and from there it is picked up by journalists so it can appear in the news. However, before it makes it there and reaches a wider audience, it undergoes the process of verification, an examination of the truthfulness of the claim, as it is important for professional journalists to deliver correct information to the public.

Over years of social media development journalists have developed common verification practices that help them identify accurate content. The verification handbook (Silverman, 2013) gives an overview of the best practices currently adopted by journalists, concentrating on verifying User Generated Content (UGC) in social media, and provides examples of use cases where each of them are applicable. Verification practices differ from journalist to journalist and between different news companies (depending on their principles and resources, and leading to their general level of reliability). This verification handbook highlights the importance of having pre-defined strict verification procedures.

Journalists monitor Twitter, especially the accounts that they find trustworthy, in order to be among the first ones to have access to fresh, newsworthy content. One of the tools used for that purpose is TweetDeck, where one can build Twitter

lists ahead of time for specific uses, for example law enforcement of major cities, reliable local reporters and the news organizations of major cities, and specialized reporters.

Deliberate hoaxes can be planted using fake tweets created through the website letmetweetthatforyou.com, presenting fake information as a retweet and adding a fake verification mark to your profile picture.

According to the verification handbook (Silverman, 2013), when a journalist comes across a piece of information on social media there are four elements for them to check and confirm:

1. **Provenance:** Is this the original piece of content?

   For content like a video or photo, one of the first questions is whether this is the original piece of footage or picture. One can find out whether it has been posted online previously by using reverse image search tools, such as TinEye or Google Image search. It is still important to remember that an image may be authentic, but it could be inaccurately labelled.

2. **Source:** Who uploaded the content?

   Journalists perform multiple checks to determine whether the account which uploaded the content in question is real. They review all of the profile information including linked websites, location, previous posts, pictures and videos, profiles of friends and/or followers. Usually journalists verifying UGC aim to determine who is the original uploader and contact them. To find this user, journalists find clues in the linking user's profile from different social networks, and check the URL via WHOIS query to find the person's address, email and personal telephone number. Once the user is identified, the journalist asks questions about the details of the event in question, such as where they were standing when the footage was taken, what could they see, and what kind of camera did they use. Using these questions journalists either convince the user to confess to content falsification or cross-reference answers with available information by examining the Exchangeable Image File Format (EXIF) metadata in a photo, or comparing footage with a specific location to Google Street View.

3. **Date:** When was the content created?

   Identifying the true date of the creation of multi-media content is not easy. Sometimes even upload date can be an indicator of falsified content, for exam-

ple if this date is earlier than the event date. Weather information may also help ascertain the date.

4. **Location:** Where was the content created?

   Very few social media posts contain geo-tags. In order to identify the location of posts without geo-tags, tools like Google Maps, Google Earth and Wikimapia can be used. Comparing the following cues with elements of the photo or video can be useful: license plates on vehicles, weather conditions, landmarks, clothing, signage/lettering, identifiable shop or building and terrain/environment.

Recent approaches include crowd-sourced verification. Also, journalists try to provide advice and guidelines to activists that are providing content, such as placing a current newspaper with the date on it in the frame of the photo.

In the following sections we focus on machine learning approaches to tasks from the rumour resolution pipeline as described in section 2.2.1 that often take inspiration from the journalist approach, for example incorporating the important factors described above as input features to the model and then testing their performance. The aim of those approaches is not to replace, but to be an aid to humans, interested in verifying claims, and to combine the ability of computers to process vast amounts of information in a short time with the intuition, experience and professionalism of a human.

2.2.3 Rumour detection task

As we have mentioned above, there is variability in the definition of rumour detection and what it entails. Often rumour detection is defined as the identification of posts that are relevant to certain predefined rumours (Qazvinian et al., 2011; Hamidian and Diab, 2015, 2016). These works describe classification models that classify the relevance of a Twitter post to the rumour, known *a priori*, and include rather long-standing rumours, such as 'Is Barack Obama muslim?' or 'Cell phone numbers going public?', however they do not include a discovery of new rumours. Thus, using the set of definitions that we follow (as outlined in section 2.2.1), this task is closer to the rumour tracking task.

In this thesis rumour detection is defined as determining whether a certain post is a check-worthy rumour (that is, its veracity is unknown, however it is verifiable and spreading widely) and not a chat or opinion piece. One of the first approaches was introduced by Zhao et al. (2015), who built a rule-based approach

to identify scepticism among responses, and therefore determine that the associated information is a rumour. The limitations of this approach consist in having to wait for responses to arrive, as well as lack of generalisability due to manually-defined rules.

Kwon et al. (2013) studied rumour detection on Twitter by exploring the predictive power of temporal, structural, and linguistic feature sets. The temporal features describe rumour spread over time. The structural features model the connectivity between users who posted about the rumour. The linguistic features were obtained using the Linguistic Inquiry and Word Count(LIWC) dictionaries (Pennebaker et al., 2001). Their comparison of Decision Tree, Random Forest, and SVM classifiers has shown Random Forest to perform better on the rumour detection task. They identified structural and linguistic differences in the spread of rumours and non-rumours highlighting that rumours show 'bursty' fluctuations over time and that the temporal feature had the highest predictive power. Further, Kwon and Cha (2014) were modelling the 'bursty' temporal pattern of rumours using agent-based modelling. They studied the concept of exclusivity, which describes the informational value of a given piece of information. Each claim has an exclusivity value, which decays over time at a certain rate. For rumours the decaying rate is smaller than non-rumours, indicating a much longer life time for rumours than non-rumours. They showed that the varying decaying rates of informational value play a significant role in the characteristic temporal pattern of rumour spread.

Qin et al. (2016) detect rumours on the Sina Weibo platform using the concept of novelty. They compare the published posts to existing news reports on the news wire to gauge the novelty with respect to the confirmed information available from trusted sources. They also assume documents, that are similar to previous rumours, to be rumours as well. Using an SVM classifier they show that novelty-based features perform well when detecting rumours instantly after their publication.

Zubiaga et al. (2017) proposed a sequential approach to leverage context from earlier posts during an event. The sequential approach achieved significant improvements over single tweet baselines, especially in terms of recall, where the rule-based approach proved limited. Tolosi et al. (2016) attempted to identify a topic-agnostic set of features that would be indicative of a rumourous story. They examined user ID, user profile, text style and URL domains, as a basis for prediction. However they found that even those features changed drastically across events, as different topics attract different public, hence making it difficult to identify rumours. Ma et al. (2018a, 2016, 2017) in their works combine the detection and verification tasks in a single 4-way classification task that includes classifying conversations into

non-rumour, true, false or unverified rumour.

The task of rumour detection is closely linked with the event (Atefeh and Khreich, 2015) and news story detection (Liu et al., 2016, 2017) tasks. For instance, Mathioudakis and Koudas (2010) proposed Twitter Monitor, an online monitoring system that performs trend detection over the Twitter stream. Twitter Monitor detects sharp increases ("bursts") in the frequency of sets of keywords found in posts and provides a relevant keyword-based query. Such a system does not identify rumours, but its output can be used as an input for a rumour detection system. Castillo et al. (2011) used Twitter Monitor as input to a model that classifies trending stories as either news or chat. Thomson Reuters Tracer system (Liu et al., 2016, 2017) is a combination of several tasks in one pipeline. Tracer detects trending stories, clusters related tweets into an event and then event summarisation, topic classification, newsworthiness ranking and finally, event verification are performed. However as Tracer is a closed-source commercial tool, the methodology and implementation details are not revealed.

Check-worthiness of claims can also be used in the context of political claims. A recent shared task (Nakov et al., 2018; Elsayed et al., 2019) aimed to predict which claims in a political speech or debate should be prioritised for fact-checking. The goal was to produce a ranked list of its sentences based on their worthiness for fact checking.

2.2.4 Rumour stance classification

Stance classification and other NLP tasks

Stance classification is the task of determining the attitude of the author of a text towards a target (Mohammad et al., 2016). Targets can range from abstract ideas, to concrete entities and events as well as different discourse segments. Stance classification is an active research area that has been studied in the context of different domains such as: congressional debates (Thomas et al., 2006), online forums (Ranade et al., 2013), bulletin board discussions (Chuang and Hsieh, 2015) and, more recently, in social media platforms (Mohammad et al., 2016).

The stance classification task has similarities with other Natural Language Processing tasks, such as sentiment analysis and recognising textual entailment. Sentiment analysis is the task of identifying, extracting and quantifying affective states and subjective information from the text. For example, in the simplest case, the task can be defined as automatically labelling certain text as positive, negative or neutral. Sentiment analysis is related to stance classification as both of the tasks look

at the author's personal, subjective information, emotion (sentiment) or attitude (stance). Stance classification always requires a target, which is not necessary for sentiment analysis. However, there are variations of sentiment analysis tasks such as target-dependent sentiment analysis (Pergola et al., 2019; Wang et al., 2017) that also, similarly to stance, require targets,which might or might not be explicitly mentioned in text. Thus, while being similar tasks that are defined with respect to a target topic, stance can be independent of whether positive or negative language was used (Mohammad et al., 2017). Sobhani et al. (2016) study the relations between the tasks and show that while sentiment features are useful for stance classification, they alone are not sufficient.

Recognising textual entailment (RTE) is the task of classifying the directional relation between text fragments, one of which is called a premise and the other a hypothesis, to determine whether the meaning of the hypothesis is entailed, i.e. can be inferred from the meaning of the premise as would typically be interpreted by people (Dagan et al., 2005). The types of relations can either be entailment, contradiction (that is, the hypothesis contradicts the premise) or neutral (neither an entailment, nor contradiction, relation between the premise and hypothesis). RTE is similar to stance classification as it is concerned with determining the relation between two instances, however in the case of RTE both hypothesis and premise are a full sentences, while stance can be determined towards any concept. These tasks can complement each other when addressing certain problems.

Rumour stance classification

One of the characteristics of rumour spread on social media is that the rumour is surrounded by user discussion of its truthfulness, which potentially leads to revealing its veracity. Rumour stance classification is the task of identifying the attitude of users participating in conversations discussing rumours towards the truthfulness of the rumour.

Rumour stance classification is also related to the task of recognising the relation between a claim and the news articles discussing it. For example, it has been studied as part of a Fake News Challenge[7] that aimed to identify whether the title of an article *agrees*, *disagrees*, *discusses* or is *unrelated*, to the suggested body of the article (Riedel et al., 2017; Hanselowski et al., 2018). In this work we are mainly concerned with stance expressed in conversations among users discussing a rumour.

[7] `http://www.fakenewschallenge.org/`

The stance can be classified on different granularity scales: binary – supporting versus denying, or three class – supporting vs denying vs none. Lendvai and Reichel (2016) performed stance classification between pairs of posts in search for contradictions, viewing it as a 3-way recognising textual entailment task with entailment, contradiction and unknown classes. They hypothesise that the amount of contradictions between posts about a rumour is linked with its veracity value. They utilise similarity features derived from the string and part-of-speech level in order to tackle the noisy text of tweets. In this thesis we follow the definition of stance classification as a 4-way classification task, as proposed by Zubiaga et al. (2016b), into *supporting* (author of the response agrees with truthfulness of the rumour), *denying* (author of the response denies the truthfulness of the rumour), *querying* (author of the response questions the truthfulness of the rumour) or *commenting* (author of the response expresses opinion that is irrelevant to the truthfulness of the rumour).

Previous research (Mendoza et al., 2010; Zhao et al., 2015; Procter et al., 2013) has shown that rumours that are later proven to be false tend to spark significantly larger numbers of denying tweets than rumours that are later confirmed to be true. Therefore patterns of support and denial towards rumours can be indicative of the final rumour veracity. We see this feature as particularly important because rumour verification models have to be able to generalise well to tackle rumours about many different topics and stance patterns could be identified irrespective of the topic, thus helping to overcome this challenge.

However the proportions of supporting and denying posts in false and true rumours differs across the published datasets. While the dataset in Mendoza et al. (2010) shows that over 60% of the tweets are either denying or questioning the rumour, Zubiaga et al. (2016b) show that true rumours tend to have only a slightly higher ratio (still statistically significant) of support than false rumours before the resolving tweet appears. They also note that while rumours remain unverified, the overall tendency is to support them, with the support ratio decreasing after the resolution of a rumour as either true or false. Researchers observed that the posting of the resolving tweet produces a higher number of tweets discussing the veracity of the rumour, either positively or negatively, especially for false rumours. Users are active at denying already debunked rumours, but are not so good at handling unresolved rumours. When it comes to true rumours, users do quite well (via expressing stance) in determining that an unverified rumour is true. Dang et al. (2016) have studied user behaviour on Reddit pertaining to rumour spread and identified three distinct groups of users: those who generally support a false

rumour; those who generally refute a false rumour; and those who generally joke about false rumours. A separate line of work (Nichols et al., 2016; Balestrucci et al., 2019a,b) studies the gullibility of users, i.e. how prone users are to believe and spread false information, and often find a connection with follower/following ratio or the types of connections user have, e.g. with online bot accounts.

Opinion is often formed or changed during a conversation, especially if the topic of the conversation is related to a recent event or exposes a user to a new idea. This suggests that the task of identifying the stance of each post in a conversation may exhibit a sequential nature. Stance classification has been considered as individual post classification, as well as a sequential problem in recent works. We describe works related to both approaches below.

Single Tweet Stance Classification

Stance classification was the focus of SemEval-2016 Task 6 (Mohammad et al., 2016) semantic evaluation task. It consisted of two subtasks. Subtask A involved judging the stance of tweets towards one of five controversial topics or public figures, namely 'Atheism', 'Climate Change is a Real Concern', 'Feminist Movement', 'Hillary Clinton', and 'Legalization of Abortion', either as 'favor', 'against' or 'none', and provided data annotated for stance. Subtask B involved stance detection towards 'Donald Trump', without the provision of data annotated for stance towards the target. Many systems obtained additional training data or weakly labelled data with their stance using manually defined heuristics (Augenstein et al., 2016). This was possible because stance targets were longstanding topics. As the corresponding dataset consists of single tweets, participants did not have the opportunity to analyse relations between tweets. However, several systems (Du et al., 2017; Augenstein et al., 2016) benefited from incorporating target information as an input to the model.

Rumour stance classification was pioneered by Qazvinian et al. (2011). Their work classified tweets related to long-standing rumours into two categories: *supporting* or *denying* the rumour. They have used different Naive Bayes classifiers as high level features, with a L1-regularized log-linear function of these classifiers, for tweet classification on various feature sets: n-grams, part-of-speech tags, user network, hashtags and URL n-grams. The model was trained on previous tweets about the rumour in question. Zeng et al. (2016) performed stance classification for rumours emerging during crises, also training on tweets involving the same rumour used during testing. This approach makes the task easier by providing a domain vocabulary, however the results do not show how this would generalise to unseen rumours and

it is not applicable at the early stages of rumour development.

Sequential Stance Classification

Lukasik et al. (2016) and Zubiaga et al. (2016a) consider the sequential nature of tweet threads in the task of rumour stance classification, albeit following different approaches. Lukasik et al. (2016) employ Hawkes processes to classify temporal sequences of tweets. They show the importance of using both the textual content and temporal information about the tweets, disregarding however, the discourse structure.

Zubiaga et al. (2016a) tackle this task proposing two ways of onversational structture composed of source tweets and subsequent replies: as a linear chain and as a tree. They use Linear- and Tree- versions of a CRF classifier, outperforming the approach by Lukasik et al. (2016).

Pamungkas et al. (2019) perform stance classification using conversation-based and affect-based features, covering different facets of affect using a SVM classifier with Radial Basis Function (RBF) kernel, without modelling the conversation structure directly by the classifier.

In this thesis we test the sequential approach to rumour stance classification and find it beneficial when compared to approaches processing tweets individually or in pairs (shown in chapter 4).

2.2.5 Rumour verification task

Rumour verification is a complex task that has multiple facets. Rumours spread on different online platforms and span many different topics, from celebrity gossip to political propaganda, thus attracting different audiences that participate in discussion of a rumour. This makes it very challenging to automate the process of verification.

One of the most promising research directions is the use of machine learning and natural language processing algorithms to tackle rumour verification.

Rumour verification implies resolving the veracity value of a rumour. The outcomes of the process could be binary, i.e. just identifying false rumours (Seo et al., 2012), or be split into three categories: true, false or unverified (Zubiaga et al., 2016b). There is also research splitting the output into six (Wang, 2017) and more (Roitero et al., 2018) categories. Wang (2017) released a dataset of claims taken from politifact.com that separates them into 6 categories: pants-fire, false, barely true, half-true, mostly-true, and true. By contrast Roitero et al. (2018)

perform claim annotation using a hundred levels of truthfulness, as well as an unbounded continuous scale.

Most commonly rumour verification is viewed as a classification problem that involves training a model on a labelled dataset in order to find patterns that characterise true and false rumours. There are very few works that propose an unsupervised approach. One of them is work by Chang et al. (2016) who proposed a rule-based method for detecting political rumours on Twitter based on identifying extreme users. They used clustering methods to identify news tweets, then employ structural and timeline features to detect extreme users. Then the truthfulness of each cluster is determined by the proportion of extreme users.

In this thesis we take a supervised learning approach. We define verification as a classification task, where given the claim in the form of social media posts, potentially with multimedia content and additional related sources, we wish to determine to which class it belongs, either *true*, *false* or *unverified* (i.e. the truthfulness is still to be identified). Ideally, a classifier is able to, together with veracity label, output its confidence in the assigned label, which can be an important aspect of interpretation of the model's results.

The main goal is to find patterns in the posts discussing rumours that are indicative of their veracity. Thus the majority of work, particularly early works, follow a similar approach. Starting with a collection of social media posts (Twitter and/or Sina Weibo) related to rumours, annotated by either relying on news articles from reputable sources or debunked by specialised web-sites like snopes.com, these works focus on identifying useful features for the classification task in combination with one of the standard classifiers (SVM, Random Forests, Naive Bayes, Logistic regression).

A lot of studies take inspiration from prior work on credibility detection and use similar feature sets, as well as implementation of those models as baselines (Castillo et al., 2011). Alrubaian et al. (2017) have collected a list of features commonly used, and deemed helpful, in the field of rumour verification (using the broad definition that includes credibility assessment and rumour detection). Figure 2.14 shows a plot from Alrubaian et al. (2017) that provides a summary of features used, and found to be useful, in a number of publications about veracity/credibility of information. According to this study, features that reveal information about the user, as well as the connectivity network between users are often shown to be helpful, therefore there is a separate line of work that concentrates on user credibility.

Also, naturally, many of the studies take inspiration from the work of human professionals tackling this task, journalists and fact-checkers. For example, Liu

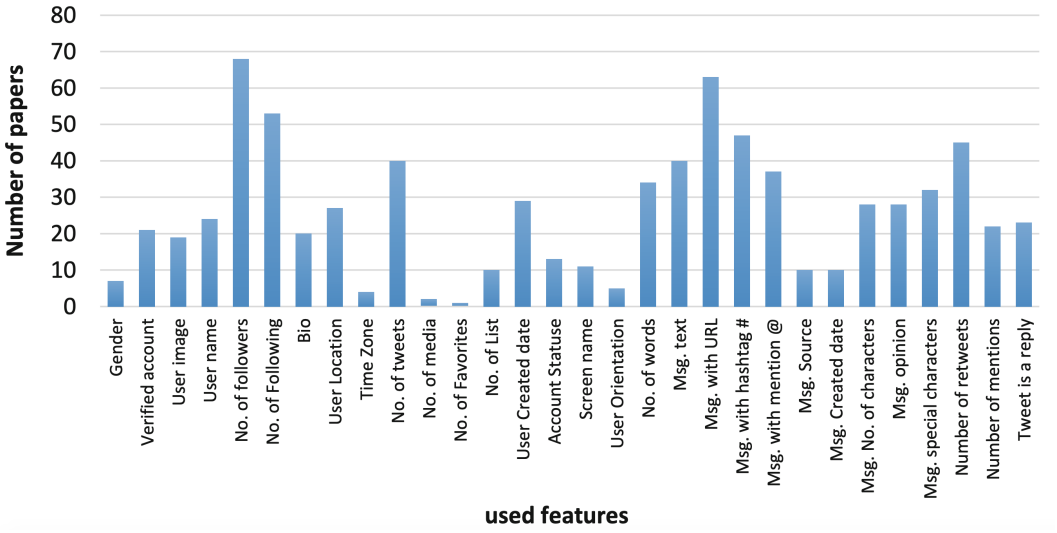


Figure 2.14: Features that are used, and considered useful, in publications on rumour credibility/veracity classification (Alrubaian et al., 2017).

et al. (2015) proposed a system that used verification features, which were determined based on insights from journalists. They included source credibility, source identification, source diversity, source and witness location, event propagation, and belief identification. In belief identification, results of rumour stance classification were used as features. The experiments were performed on the authors own dataset using SVM classification. The results have shown that features based on crowd stance towards rumours helped improve model performance.

Even though there are a lot of rumours and misinformation being spread online, rumourous posts are still a minority compared to non-rumours (including any chat, opinion and even verified news). Therefore, a lot of works choose to balance the classes in the datasets containing rumours when collecting the data (Kwon et al., 2013; Wu et al., 2015). Differently to other works in the field, Chen et al. (2016) proposed to tackle the task of rumour verification on Sina Weibo as binary classification from the perspective of anomaly detection, where false rumours are viewed as anomalies. The authors use Factor Analysis of Mixed Data (FAMD), which they describe as a combination of PCA (Principle Component Analysis) and MCA (Multiple Correspondence Analysis), to detect these anomalies.

Two shared tasks attracted further attention to the problem of rumour verification: RumourEval 2017 (Derczynski et al., 2017) and RumourEval 2019 (Gorrell et al., 2019). These provided participants with datasets of Twitter conversations discussing rumours annotated for stance and veracity classification tasks.

In RumourEval 2017 (Derczynski et al., 2017), veracity classification was viewed as 3-way single tweet classification task (true, false, unverified) as the participating teams did not make use of the whole conversation provided. Most of the submitted systems chose the closed variant of the task, with no external resource use permitted. The winning system NileTMRG (Enayet and El-Beltagy, 2017) used a SVM-based model and utilised stance as an indicator of veracity (described in chapter 5).

In RumourEval 2019 (Gorrell et al., 2019), submitted systems showed a strong trend for neural approaches. The winning system eventAI (Li et al., 2019b) implemented an ensemble of classifiers: SVM, RF, LR and a neural network, where individual tweet representations are created using an LSTM with attention (Xu et al., 2015; Rocktäschel et al., 2015).

In the remainder of this section we present an overview of different types of supervised learning approaches to rumour verification. Many studies on rumour verification consider linguistic features and lexical cues, others consider information about the user network and rumour propagation through it, as well as information about users who spread and respond to rumours. Some works pay attention to the temporal and structural information of conversations discussing rumours, as well as the reaction of the public to the rumour. In order to augment the texts of social media posts some works use any attached images and external news articles. Often works combine several types of approaches, therefore the division presented here is not mutually exclusive. The works in each of the following sections are roughly organised by publication year, and include some of the recent works that were released after the work described in this thesis. In this thesis we focus on utilising the information contained in a post conveying a rumour, extracting potentially relevant features such as linguistic markers, user information and its social interactions; as well as processing the conversation around the tweet and gathering the stance of responses towards a rumour.

Approaches using linguistic features

Many studies on rumour verification start by examining linguistic features and lexical cues. Zhang et al. (2015) studied health rumours using logistic regression. They concluded that the lengths of rumour headlines and statements, and the presence of pictures, are negatively related to the probability that a rumour is true, while a rumour is more likely to be true if it contains elements such as names of people or places, numbers, source cues and hyperlinks.

Chua and Banerjee (2016) also studied linguistic predictors of rumour veracity on the internet, which included (1) comprehensibility, (2) sentiment, (3) time-orientation, (4) quantitative details, (5) writing style, and (6) topic using the Linguistic Inquiry and Word Count (LIWC) (Pennebaker et al., 2001). They found that rumour feature types (1)–(4) were indicative of rumour veracity. Negatively-phrased rumours were more likely to emerge as being true compared with affirmative rumours. Rumours rich in past tense with a rare use of present and future tenses were likely to be true. Rumours rich in discrepancy and swear words but sparse in terms of exclusion words were likely to be true. Rumours that used words on non-controversial topics, such as home and leisure, were likely to be true. On the other hand, rumours that were rich in words on controversial topics, such as religion and sex, were likely to be false. Interestingly, it was found that the use of sentiment and quantitative details in rumours could not predict their veracity. This finding contradicts prior works, such as Kwon et al. (2013) (on rumour detection), which found sentiments useful in predicting rumour veracity on Twitter, as well as those like Zhang et al. (2015), which deemed the use of quantitative details a crucial predictor of veracity for health rumours.

Reichel and Lendvai (2016) considered lexical cues, as well as perceived certainty trends, to determine the veracity of a set of rumourous claims on Twitter. They tackled two tasks: rumour veracity classification into *true* and *false* classes, and the identification of a resolving tweet among tweets in a conversation. The results suggested that the lexical cue set was helpful in identifying the resolving tweet, while the certainty features turned out to be predictive of rumour veracity.

Yang et al. (2012) tackled the veracity of microblogs on Sina Weibo. The authors adopted features that were useful in earlier studies, such as Castillo et al. (2011), which include features based on language, user, propagation, and other meta data, and extended them with two more features: client-based and location-based features, namely the client program used and the event location. Authors showed that adding the two features on top of the propagation-based features reported by Castillo et al. (2011) led to a significant performance increase.

Vosoughi (2015) studied veracity classification on Twitter datasets gathered by the author. They used linguistic, user oriented, and temporal propagation features. However, the best performing features were those in the temporal propagation category.

The above-mentioned studies were performed on different datasets and in different model evaluation scenarios. Unfortunately, they do not present us with a consistent pattern of linguistic features for identifying a rumour, and in fact some of

the findings contradict each other. This shows that neither of the datasets is fully representative of the studied phenomena, as rumours cover a very diverse range by topics and are conveyed by different people. While this absence of obvious pattern does not prove that there is no existing pattern, it does suggest the importance of searching for topic- and vocabulary-independent features indicative of rumour veracity.

Network/User approaches

There is a prominent line of work focusing on the users spreading rumours and their connection network. Mendoza et al. (2010) explore the behaviour of Twitter users under an emergency situation. They use a dataset of tweets related to the 2010 earthquake in Chile, which contains true and false rumours. They analysed the social network of the community surrounding the topic and show that the propagation of tweets that correspond to false rumors differs from the true ones because false claims tend to be questioned more than news by the users.

Seo et al. (2012) study false claims and true information on Twitter. They model the social network as a directed graph, where vertices represent individuals and directed edges represent information flow via follower-followee relations. The authors inject the network with monitor nodes who report the data they receive. The proposed algorithm identifies rumours and their sources by observing which of the monitors received the given piece of information and which did not. Results show that with a sufficient number of monitor nodes, it is possible to recognise most rumours and their sources with high accuracy.

Yang et al. (2015) made use of the comments attached to the source tweet, which is spreading the rumour. They have proposed the use of network features, which were derived from a social network created from users who leave comments, to perform the rumour veracity classification task. They have shown that when the network feature was added to the traditional features, the results of classification improved substantially.

Vosoughi (2015) has compared effects of several feature types (linguistic, user and propagation dynamics over the network) extracted from a rumour timeline over a model's performance. Propagation features that have significantly contributed to the outcome of the models were (1) fraction of low-to-high diffusion, (2) fraction of nodes in largest connected component (LCC), (3) average depth-to-breadth ratio, (4) ratio of new users, ratio of original tweets, (5) fraction of tweets containing outside links, and (6) the fraction of isolated nodes. All of these features are derived from a rumours diffusion graph. The study found that the inclusion of features in the

temporal propagation category led to the best performing systems. In disagreement with some of the earlier findings, Vosoughi (2015) found that user-oriented features did not contribute as much as other feature categories in their experiments.

Wang and Terano (2015) proposed social graphs to model the interaction between users, thereby identifying influential rumour spreaders. The graph entailed information about familiarity, measured by the number of contacts, such as RTs, replies, and comments between two users, activeness measured by the number of days a user has sent out messages, similarity measured by gender and location, similarity between two users, and trustworthiness measured by whether the user is verified or not. These four factors were merged in a linear model, and hence the model was used to weight the link between two users in the social graph. Influential spreaders were used to determine rumours.

Chang et al. (2016) determined the truthfulness of clusters of tweets through evaluating the proportion of extreme users in them that are identified using structural and timeline features of a rumour.

The above studies represent a fruitful and important line of work that utilise the connections and interactions between users of a social media platform in order to identify rumours and their veracity. In this thesis we also consider user features (using information from user profiles) and social interactions (in the form of number of likes and shares), however we do not focus on processing rumour propagation networks as this calls for expertise from the network theory field.

Temporal and structured approaches

Further developments in the field of rumour verification led researchers to focus on the temporal component of rumour spread, treating rumour verification as a time-sensitive task, as well as on the context provided by the conversation around the rumours (structural). Temporal or structural components can be expressed as features or as part of the model architecture.

Temporal or structural components expressed as features. Works by Kwon et al. (2013); Kwon and Cha (2014) on rumour detection detection have explored the temporal patterns of rumour spread and attempted to distinguish it from the spread of non-rumours by characterising its 'bursty' patterns.

In later works Kwon et al. (2017) considered that rumours are claims that are either resolved as unverified or false, hence attempting the verification of the information. They compare classification model performance changes over varying time windows. They have examined the contribution of user, structural (structural

properties of the rumour diffusion network), linguistic, and temporal features. Temporal features were extracted from daily time series of the number of tweets. They find that time series of tweet numbers show multiple and periodic spikes for rumour events, while most non-rumour events appear with a single prominent spike. The results show that while temporal features distinguish rumours from non-rumours over a long-term window, they are not available during the early stages of propagation. On the other hand, user and linguistic features are available instantly and also act as a good rumour indicators. This highlights the need for an approach that utilises a combination of different types of rumour markers.

Giasemidis et al. (2016) measured trustworthiness of a claim at varying time windows. They split every rumour into 20 time-intervals and extract all the features for each subset of tweets. Comparing several classifiers (Logistic Regression, linear kernel SVM, RBF kernel SVM, Random Forest, Decision Tree, Naive Bayes) this study found Decision Trees, Random Forest and Logistic Regression to work best. Their models reach 76% accuracy at one quarter of the rumour duration, with accuracy increasing over time with more tweets and information becoming available.

Temporal or structural components expressed as part of model architecture. Similarly to Kwon et al. (2017), Ma et al. (2016) considered that rumours are claims that are either resolved as unverified or false. They have performed classification of posts from Twitter and Sina Weibo into rumours and non-rumours using Recurrent Neural Networks. Their results show the superior performance of Gated Recurrent Unit (GRU) -based (Cho et al., 2014) networks over the non-sequential approaches using SVM and Random Forest classifiers, as well as a sequential LSTM-based approach.

Chen et al. (2017) also used recurrent neural networks (LSTM) which they enriched with a neural attention mechanism in order to classify posts into false rumours and non-rumours, focusing on the possibility of early detection. The neural attention mechanism (Xu et al., 2015; Rocktäschel et al., 2015) is used for highlighting the relevant part of rumours that can lead to its debunking. The proposed approach outperforms the previous baseline approaches, including Ma et al. (2016).

Vosoughi (2015) have modelled time series of several feature types (linguistic, user and propagation dynamics over the network) extracted from a rumour timeline using Dynamic Time Wrapping (DTW) and Hidden Markov Models (HMMs) thus making the use of the temporal component of rumour resolution. Comparison of the classifiers showed that HMMs outperformed DTWs.

Wu et al. (2015) extracted features from message propagation trees. They proposed a graph-kernel-based hybrid SVM classifier, which captures the high-order propagation patterns in addition to semantic features, such as topics and sentiments. Three categories of features were considered: message-based, user-based, and report-based. Their results show that the repost patterns of false rumours are very different to non-rumours (described as 'normal' messages that were not proven to be false), which makes the graph kernel very useful in detecting false rumours. The combination of random walk kernel and RBF kernel performs better than each of them alone.

Ma et al. (2017) also performed classification of claims spreading on Twitter into 4 categories: non-rumour, true, false and unverified. They modelled propagation structure through kernel learning. They proposed two models: a novel PTK kernel that is able to capture the structure of a conversation propagation trees and its cost-sensitive version cPTK. Their PTK and cPTK models were shown to ourperform all other existing baselines, including Ma et al. (2016).

Ma et al. (2018a) continued experiments with a 4-way classification and proposed two neural models that are able to represent the tree structure of a propagation tree. One of the models is a Top-Down Tree GRU that represents information flow from the source posts to the leaf nodes, and the second one is Bottom-Up Tree GRU that models the opposite information flow. Experiments show that Top-Down Tree GRU outperforms previous baseline approaches and Bottom-Up Tree GRU.

Dungs et al. (2018) investigated whether rumour stance combined with tweet times can be used to predict rumour veracity. They modelled the veracity of a rumour using variants of Hidden Markov Models (HMM) with collective stance information, and showed that their approach outperforms the stance unaware baselines and the stance-aware model of Liu et al. (2015).

The most recent work is by Kumar and Carley (2019) who combined tree-structured LSTMs with convolution layers to predict the stance towards a rumour and its veracity in social media conversations on PHEME dataset. The tree structures in this work represent the structure of a conversation surrounding a rumour.

In this thesis we also investigate the effect of incorporating conversation structure that also preserves some temporal links between the posts, as applied to stance classification and rumour verification models to improve their performance.

Stance

As a lot of work in this thesis concerns stance classification and relation of this task with rumour verification, we highlight works that also utilised and benefited from

this connection.

Liu et al. (2015) proposed a system that outperformed the systems proposed by Yang et al. (2012) and Castillo et al. (2011). Liu et al. (2015) used verification features determined based on insights from journalists, and included source credibility, source identification, source diversity, source and witness location, event propagation, and belief identification. In belief identification, results of rumour stance classification were used as features. The experiments were performed on the authors own dataset using SVM classification. The results have shown that features based on crowd stance towards a rumour helped improve the model's performance.

Enayet and El-Beltagy (2017) won the RumourEval 2017 shared task with their system that incorporated a two-step approach to rumour verification. First they label the stance of each of the responses towards the rumour, then use the outcome of this step for rumour verification. They propose to use a linear SVM model, with source tweets represented as bag-of-words, with extra features indicating the presence of a hashtag and proportion of supporting, denying and questioning stances towards a rumour amongst the responses.

These works by Liu et al. (2015) and Enayet and El-Beltagy (2017) inspired our investigation of automated stance classification systems and the relationship between the stance classification and rumour and verification tasks in chapters 4 and 5. The works by Dungs et al. (2018) and Ma et al. (2018b) are contemporary to the work presented in chapter 6 and also support our findings of the usefulness of the stance of responses for identifying rumour veracity. Dungs et al. (2018) showed that the distribution of stances in the rumour timeline processed by Hidden Markov models leads to positive outcomes on the partial PHEME dataset. Ma et al. (2018b) proposed to classify rumour veracity and stance jointly by neural multitask learning using stance posts from RumourEval 2017 and rumour propagation trees.

A more recent work by Kumar and Carley (2019) also took a multitask learning approach but with a different model architecture, using tree LSTMs with CNNs to predict stance and rumour veracity for the PHEME dataset.

Media and images

Often rumours are spread in the form of fake images, images that were digitally manipulated or genuine images with the wrong captions, mis-representing the situation. Gupta et al. (2013) focuses on identifying fake images. This is in line with research on the image forensics field, which studies identifying whether the image has been tampered or altered. Usually such software looks for anomalies in RGB colour space or other colour spaces, such as discrete changes in a level of colour to

determine if pixels were manipulated. Gupta et al. (2013) performed analysis of the temporal, social reputation and influence patterns for the spread of fake images. The vast majority of tweets spreading the fake images were retweets, hence there were very few original tweets. They performed classification of images from Hurricane Sandy to distinguish between the fake and real ones using a Decision Tree algorithm. Tweet-based features were very effective in distinguishing fake images tweets from real, while the performance of user-based features was rather poor.

Boididou et al. (2014, 2017) have also focused on tweets that are related to fake images, bringing a multimedia component into the verification process. Boididou et al. (2014) have created a dataset of tweets around big events focusing on the ones linking to images (verified as fake or real). They classified tweets with unreliable media content as fake or real using content and user features for each tweet with J48 (an open source Java implementation of the C4.5 decision tree algorithm (Quinlan, 2014)), Kstar (Cleary and Trigg, 1995) and Random Forest classifiers. Importantly, their results have shown that using different events for training and testing, lead to much lower accuracy scores, demonstrating that the generalisation of the predictor is a very challenging issue.

In the following work, Boididou et al. (2017) proposed a new set of features and a semi-supervised event adaptation approach that helps generalise the trained models to unseen content, outperforming the baselines provided by the previous work. The system consisted of two Random Forest classifiers working on Twitter-based and user-based feature sets separately, followed by bagging (Breiman, 1996), with the final prediction calculated using a majority vote. In this thesis we do not incorporate the information from the images attached to the tweets, however this could be a line of future work.

External information

Very few works try to link rumours from social media with external information. Relevant external information can be in the form of news articles associated with the claim or some general background knowledge about the world, structured or unstructured. Qin et al. (2016) use relevant information from newswire (a service transmitting the latest news stories via satellite, the Internet, etc[8]). Boididou et al. (2017) used some features based on links attached to tweets, such as Web of Trust score and credibility ranking of the page. We believe that incorporating information from multiple sources is an important direction for future work.

[8] https://www.merriam-webster.com/dictionary/newswire

CHAPTER 3

Datasets

Chapter 2 describes previous works analysing the problem of rumours on social media platforms. Those works involve collecting and annotating datasets from those platforms with respect to the task for which they are used. The two most prominent strategies for collecting datasets of rumours are: (1) start with gathering debunked rumours on websites like snopes.com, emergent.info or politifact.com, and then collecting social media posts discussing them; (2) start with collecting the stream of social media posts related to some topic or event, and then identify the rumour stories if they are present in the collected dataset. Zubiaga et al. (2018b) refers to these strategies as *top-down* (1) and *bottom-up* (2) respectively.

The datasets produced using these collection strategies should be annotated accordingly. When the first strategy is used, the post on the debunking website is used to extract keywords, which would guide the search for relevant social media posts. The veracity label of the rumour is also provided by the debunking website, thus the annotation is only required to confirm that the collected social media posts are indeed relevant to the rumour in question and can be performed using crowd sourcing (Kwon et al., 2017). True rumours are sometimes collected using stories from the news as sources (Liu et al., 2015). If the dataset is also intended to be used for the task of rumour detection, then it still needs to be augmented with non-rumour posts that are often just collected from the social media platform application programming interface (API) stream and either added without annotation (Ma et al., 2016), or annotated as news, chat or opinion posts (Castillo et al., 2011). This type of collection and annotation is limited to the rumours which have attracted the attention of a debunking website. It also relies on the assumption that by the time a rumour is resolved by a debunking website, the social media posts are still publicly available. This is a rather common approach, used in Qazvinian et al. (2011), Liu et al. (2015) and Procter et al. (2013).

When the second collection strategy is used, the rumours are unknown in

advance, and thus a professional judgement on the presence of rumour stories and their veracity is needed. This makes the second collection strategy likely to be more expensive than the first one as professional help is required, however the resulting dataset is closer to the realistic set up and supports the collection of fast-paced emerging rumours related to breaking events. Additionally, it does not need to be augmented further as it already contains both rumours and non-rumours with a realistic class balance between them, and between true and false claims. If stance annotation of posts is needed when using either annotation strategy, it can be performed using crowd sourcing. This type of collection was used in Zubiaga et al. (2016b), Giasemidis et al. (2016) and Derczynski et al. (2017).

Early works collected sets of social media posts related to rumours without conversations around them (Liu et al., 2015; Kwon et al., 2017). Throughout the work in this thesis we have used publicly available datasets with a strong focus on Twitter conversations. Twitter is a popular microblogging and social networking platform on which users post messages with restricted length, known as "tweets". The user base of Twitter is estimated to be 330 million of monthly active users in the first quarter of 2019[1]. We used these specific datasets, namely PHEME, RumourEval Twitter 15 and Twitter 16, because their collection and annotation aligns with the aims of our project and task definitions to which we adhere. These datasets accommodate the testing of our hypothesis that the context provided by the discussion around a rumour is important for the tasks in the rumour resolution process. The datasets we have used provide the conversation around a rumour, showing the timeline of its development, allowing for time-sensitive modelling.

Additionally, we have utilised a dataset from forum debates and a dataset of rumour claims and relevant articles, both described in section 3.4 in order to test the generalisability of our models to a wider range of tasks.

3.1 PHEME

PHEME is a publicly available dataset of Twitter conversations discussing nine newsbreaking events (Zubiaga et al., 2016b). We refer to the events using identifying keywords: "Charlie Hebdo", "Ferguson", "Sydney siege","Ottawa shooting", "Germanwings crash", "Putin missing", "Prince-Toronto", "Ebola-Essien", and "Gurlitt'. Table 3.1 explains the events refered to by the keywords. The PHEME dataset contains three levels of annotation for the tasks of rumour de-

[1] https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/

Keyword	Event
Charlie Hebdo	Shooting in the offices of the French satirical weekly newspaper Charlie Hebdo in January 2015
Ferguson	Unrest that began the day after the fatal shooting of Michael Brown by police officer Darren Wilson on August 9, 2014 in Ferguson, Missouri
Sydney siege	Hostage crisis in a cafe in Sydney in December 2014
Ottawa shooting	Shootings at Parliament Hill, Ottawa in 2014
Germanwings crash	Crash of a Germanwings Flight 9525 in March 2015
Putin missing	Rumours about Russian president being missing
Prince-Toronto	Rumours about concert of Prince in Toronto
Ebola-Essien	Rumours about Michael Essien contracting Ebola
Gurlitt	Rumours about the collection of art belonging to Gurlitt being passed to the Bern museum

Table 3.1: Explanation of events in PHEME dataset.

tection, rumour stance and veracity classification [2]. First, each conversation tree is annotated as either rumour or non-rumour; second, rumours are labelled as either true, false or unverified. And third, a subset is annotated for stance classification at the tweet level through crowd-sourcing.

The data was collected as each of the events were unfolding in real time. Not just the relevant tweets provided by the Twitter API were collected, but also full conversations around them. Then professional journalist manually selected rumour stories and annotated them as true, false or unverified (Zubiaga et al., 2016b). Rumour story refers to a single claim relevant to the event; rumour story can be discussed in multiple conversations. Each of the conversations, belonging to the same rumour story gets assigned the same veracity label of the story. Conversation trees are then used as an input instances in the experiments.

The stance of the responses to the rumours can be labelled by non-expert workers as labels can be inferred directly from the text; the rumour verification task is more challenging as it requires analysis of the context, and further understanding of the rumours in order to determine if the underlying story is true, false, or remains unverified.

Figure 3.1 shows an example of a conversation discussing a rumour about Michael Essien having contracted Ebola. The conversation consists of a source tweet conveying a rumour and a tree of responses, expressing their opinion towards the claim contained in the source tweet. The veracity label of the rumour is false, while

[2] https://figshare.com/articles/PHEME_dataset_for_Rumour_Detection_and_Veracity_Classification/6392078

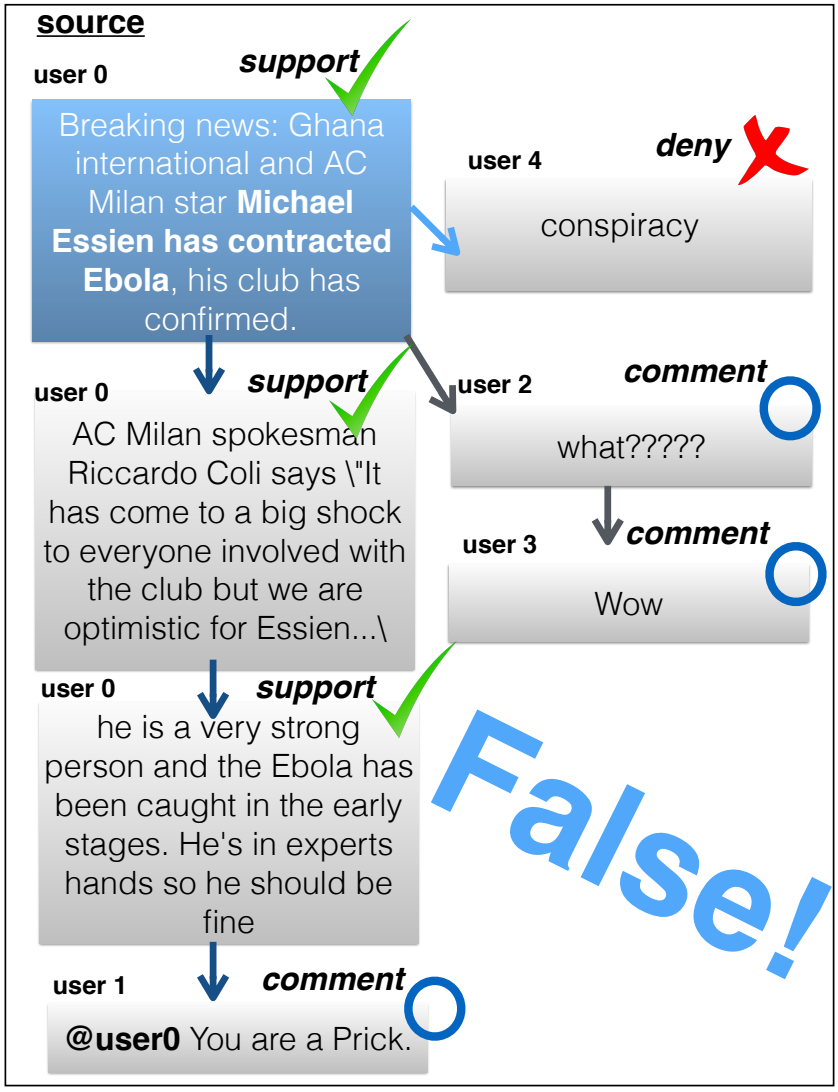


Figure 3.1: Example of a conversation with 3 branches from the PHEME dataset. Branches are indicated in the arrow color.

Events	Trees	Tweets	Rumours	Non-rumours	True	False	Unverified
Charlie Hebdo	2,079	38,268	458	1,621	193	116	149
Sydney siege	1,221	23,996	522	699	382	86	54
Ferguson	1,143	24,175	284	859	10	8	266
Ottawa shooting	890	12,284	470	420	329	72	69
Germanwings-crash	469	4,489	238	231	94	111	33
Putin missing	238	835	126	112	0	9	117
Prince Toronto	233	902	229	4	0	222	7
Gurlitt	138	179	61	77	59	0	2
Total	6,425	105,354	2,402	4,023	1,067	638	697

Table 3.2: Number of conversation trees, tweets and class distribution in the PHEME dataset.

each of the responses could be tagged as either supporting, denying, questioning or commenting on the rumour. Conversations can be decomposed into branches, such that a branch is a linear sequence of tweets starting from a leaf node of the conversation tree and going through its parent nodes up to the source tweet. The conversation on the figure 3.1 can be decomposed into three branches.

The number of rumours, the number of the corresponding conversations, as well as the class label distribution vary greatly across events. Table 3.2 shows the size of each event in the PHEME dataset as well as the label distribution for the tasks of rumour detection and verification. Tweet branches contain different number of tweets: average branch length is 2.9; minimum branch length is 2; and maximum branch length is 25 tweets. Overall, the PHEME dataset contains fewer rumours than non-rumours, while the majority veracity class for rumours is true. The information about the stance classification task is in table 3.3 (training and development) describing the RumourEval 2017 dataset, as the PHEME dataset formed a basis for the training and development sets of RumourEval 2017.

A few of the tweets contain images attached. One would assume that those would be photos from the event sites to provide evidence for the claim in question, however, after manual inspection, given the nature of events, most of the images belong into two categories: artworks conveying sentiment and screenshots of news programs. Also, tweets often contain links to other websites, such as news articles to support the claim. The PHEME dataset contains the latest revision of the linked articles prior to the link being tweeted, from the Internet Archive, where available.

To assess the difficulty of performing the verification task by a non-expert, we went through the rumours and annotated them for veracity. The overlap between the non-expert annotator and the journalist was within the range of $60 - 65\%$ on the rumour stories from the five largest events.

When conducting experiments on this dataset we perform cross-validation in

a leave-one-event-out setting, i.e. using all the events except for one as training, and the remaining event as testing. This is a challenging setup, imitating a real-world scenario, where a model needs to generalise to unseen rumours.

We used the PHEME dataset for experiments in Chapters 4, 5, 6, 7.

3.2 RumourEval

RumourEval is a shared task that was held as part of the annual SemEval competition, an International Workshop on Semantic Evaluation. At the moment of writing this thesis RumourEval has been held twice, in 2017 (Derczynski et al., 2017) and 2019 (Gorrell et al., 2019). RumourEval datasets contain Twitter conversation trees associated with different newsworthy events, including the Ferguson unrest, the shooting at Charlie Hebdo, the shooting in Ottawa, the hostage situation in Sydney and the crash of a Germanwings plane. The basis of the training and development sets of both editions of RumourEval was formed by a subset of Twitter conversations from PHEME, which was further extended. Thus figure 3.1 is representative of instances in the RumourEval 2017 and 2019 datasets. All conversations in the RumourEval datasets are rumours, thus RumourEval 2017 and 2019 consist of two subtasks: subtask A - stance classification, and subtask B - veracity classification.

3.2.1 RumourEval 2017

RumourEval 2017 contains 325 Twitter conversation trees discussing rumours consisting of 5568 underlying tweets annotated for stance at the tweet level. It is split into training, testing and development sets. The testing set contains a mix of rumours related to the same events as in the training and development sets, with the addition of two rumours: about Marina Joyce and the health condition of Hillary Clinton. Table 3.3 shows the number of conversation trees, branches and tweets in each of the sets of the RumourEval 2017 dataset, as well as the class distribution for both tasks.

In the stance classification task there is a clear class imbalance in favour of *commenting* tweets (66%) and *supporting* tweets (18%), whereas the *denying* (8%) and *querying* classes (8%) are under-represented. While this imbalance poses a challenge, it is also indicative of the realistic scenario where only a few users question the veracity of a statement. For the verification task, the training set contains more true instances than false or unverified, whereas the development and testing sets are more balanced.

	Trees	**Branches**	**Tweets**	**True**	**False**	**Unver.**	**S**	**D**	**Q**	**C**
Development	25	215	281	10	12	3	69	11	28	173
Testing	28	772	1049	8	12	8	94	71	106	778
Training	272	3030	4238	127	50	95	841	333	330	2734
Total	325	4017	5568	145	74	106	1004	415	464	3685

Table 3.3: Number of conversation trees, tweets and class distribution in the RumourEval 2017 dataset.

We have used the RumourEval 2017 dataset in Chapters 4, 5, 6.

3.2.2 RumourEval 2019

RumourEval 2017 dataset was used as training and development sets in the RumourEval 2019. The additional English Twitter testing data is about natural disasters. In such events, where chaos dominates the situation, rumours are spread on various issues and false rumours have the potential to increase the chaos. Detecting such false rumours is important to plan actions that will prevent the additional negative impact on an already existing chaotic situation. To collect this dataset rumours about natural disasters were chosen manually through Snopes.com and Politifact.com. Thus the rumours and their labels were taken from the debunking web-sites, then a search for relevant social media posts and corresponding conversations was performed. The number of source tweets of different veracity and replies of different stances are given in table 3.4. Each of the new tweets was annotated for stance towards the rumour using crowd sourcing. The distribution of stances provided for the replying tweets is shown in the table 3.4, where overall the distribution of stances is skewed towards the comment category, as was the case in RumourEval 2017 and PHEME.

One of the most significant differences between RumourEval 2017 and 2019 is the addition of rumours from a second social media platform, Reddit. Figure 3.2 shows an example of a Reddit conversation in the RumourEval 2019 dataset. Rumours were identified on Reddit by manually searching debunking and current affairs forums to identify suitable conversations. Unlike Twitter, Reddit does not have restriction on the length of the post, hence the discussions tend to be more in-depth, often with a complex conversational structure exploring the topic. They are usually introduced by a post implicitly querying the rumour, unlike Twitter rumours which are more often presented as valid information and therefore the source tweets usually support the rumour. Reddit conversations were annotated for their veracity using the debunking websites where their source was found. Posts were also annotated for stance using crowd sourcing with rigorous instructions for

	True	False	Unver.	Total Trees	S	D	Q	C	Total Tweets
Twitter Train	145	74	106	325	1004	415	464	3685	5568
Reddit Train	9	24	7	40	23	45	51	1015	1134
Total Train	154	98	113	365	1027	460	515	4700	6702
Twitter Test	22	30	4	56	141	92	62	771	1066
Reddit Test	9	10	6	25	16	54	31	705	806
Total Test	31	40	10	81	157	146	93	1476	1872
Total Task	185	138	123	446	1184	606	608	6176	8574

Table 3.4: RumourEval 2019 corpus statistics.

the annotators (Gorrell et al., 2019).

We have used RumourEval 2019 dataset in Chapters 4 and 5

3.3 Twitter 15 and Twitter 16 datasets

The Twitter 15 and Twitter 16 datasets[3] were made publicly available by Ma et al. (2017), and were created using reference datasets from Ma et al. (2016) and Liu et al. (2015).

The Liu et al. (2015) dataset consists of 94 true and 446 false stories that were taken from rumour debunking websites, namely snopes.com and emergent.info. Rumour annotation is at the event level, then all relevant tweets for each story are collected using keyword-based queries that describe each story. Additional true stories were collected using their custom event detection system, a clustering algorithm that groups tweets of the same stories or events with outcomes similar to Twitter Monitor (Mathioudakis and Koudas, 2010), and filtering those for events containing links to news that could confirm them. The resulting Liu et al. (2015) dataset consisted of 421 true and 421 false events.

Ma et al. (2016) collected confirmed rumors and non-rumors from 778 events reported on snopes.com during March-December 2015. For each event, Twitter posts were collected using keyword queries extracted from the last part of the Snopes URL. Then, to balance the classes, some non-rumor events were added from two public datasets by Castillo et al. (2011) and Kwon et al. (2013). The resulting dataset contains 498 rumors and 494 non-rumors. This dataset also has posts from the Sina Wiebo platform, a Chinese microblogging website, but this part was not used further.

Finally, Ma et al. (2017) compiled Twitter 15 and Twitter 16 datasets on the basis of the above-mentioned Liu et al. (2015) dataset for Twitter15, and the

[3]These two datasets can also be shortly referred to as Twitter 15/16.

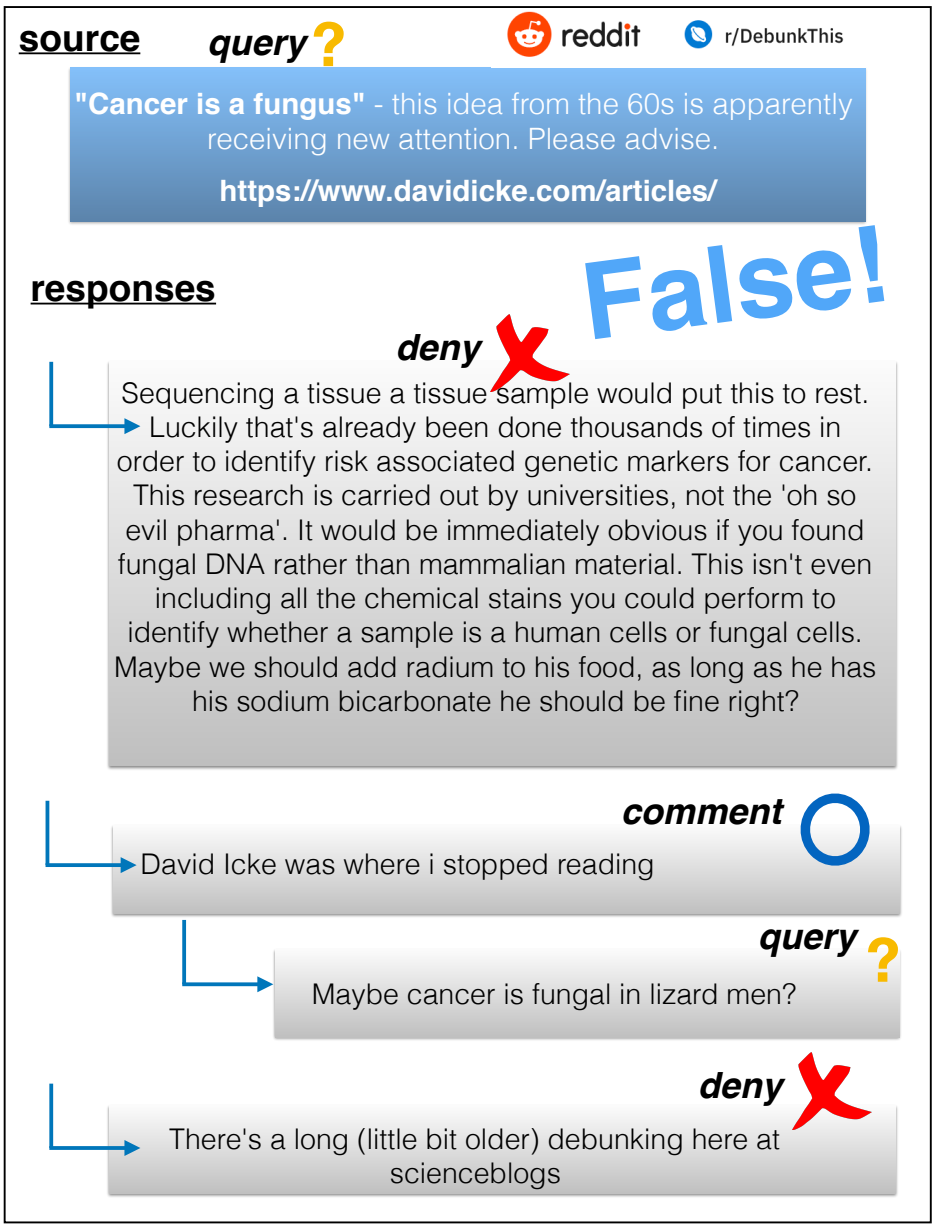


Figure 3.2: Example of a Reddit discussion from RumourEval 2019 dataset.

	# Rumours	# Tweets	True	False	Unverified	NR
Twitter 15	1374	40927	350	336	326	362
Twitter 16	735	18770	189	173	174	199

Table 3.5: Number of rumour trees and class distribution in the Twitter 15 and Twitter 16 datasets (NR – Non-Rumour).

Ma et al. (2016) dataset for Twitter16. They extracted the popular source tweets that are highly retweeted or replied to and then collected all of the propagation trees (i.e., retweets and replies) for these source tweets. Importantly, they turned the label of each event in Twitter15 and Twitter16 from binary to 4-class according to the veracity tag of the article in rumor debunking websites (e.g., snopes.com, Emergent.info, etc). These datasets merge rumour detection and verification into a single four-way classification task, containing True, False and Unverified rumours as well as Non-Rumours. Twitter 15 and Twitter 16 datasets do not contain stance annotations.

Table 3.5 shows the number of rumour conversation trees in the datasets and number of trees in each of the four classes. The dataset is made publicly available as a list of tweet IDs, connections between the tweets and annotations, therefore we had to perform the collection of tweets ourselves. Thus as we performed it later than the original dataset collection, we have only obtained those tweets that were still publicly available, losing some of the posts present in the original data. The average number of posts in a tree in the Twitter 15 dataset is 223 and 251 in Twitter 16, while the maximum number of posts in a tree in the Twitter 15 dataset is 1768 and 2765 in Twitter 16 (Ma et al., 2017).

Both datasets are split into 5 folds for cross validation, and contrary to the PHEME dataset, folds are of approximately equal size with a balanced class distribution.

We have used Twitter 15 and Twitter 16 datasets in Chapter 7.

3.4 Other

3.4.1 ABCD

The ABCD (Agreement by Create Debaters) dataset was introduced and made publicly available[4] by Rosenthal and McKeown (2015). It contains 182,308 posts coming from 12,931 two-sided debates from the 'Create Debate' forum (createdebate.com).

[4]http://www.cs.columbia.edu/ sara/data.php

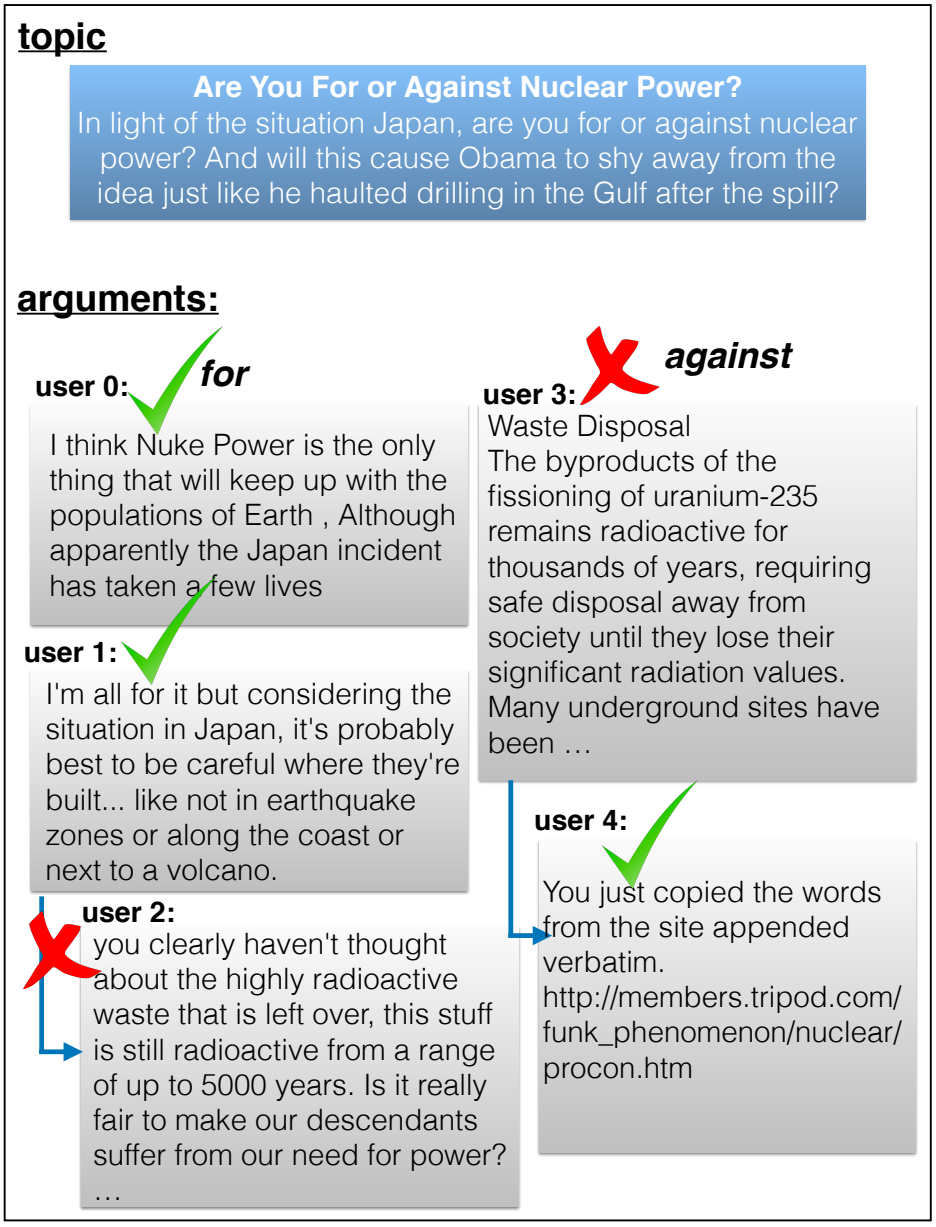


Figure 3.3: Example of a debate from the ABCD dataset.

	Topics	**Branches**	**Posts**	**Denying**	**Supporting**
Training	10,294	46,829	145,045	57,942	87,103
Development	1,315	6,053	18,641	7,567	11,074
Testing	1,322	6,066	18,622	7,556	11,066

Table 3.6: Number of topics, branches and posts in the ABCD dataset broken down by training, development and testing sets.

In this corpus, the participants in the debate choose what side they are on each time they participate in the discussion. Figure 3.3 shows an example of a debate instance from the ABCD dataset.

The authors of the dataset provide the split into training, testing and development sets. Table 3.6 shows the number of topics, branches and posts in each of the sets. The ABCD dataset is significantly larger than the PHEME datatset and the posts are longer as there is no 140 character restriction on the forum.

Original annotation was concerned with agreement between consecutive responses, i.e. if the post and a response to it were on the same side of a debate then they are in agreement, and in disagreement otherwise. In our experiments we introduced an annotation scheme to follow the same principles as in the PHEME dataset: stance is determined towards the source post of the conversation by checking whether the source post and response belong to the same side of the debate. Table 3.6 shows the resulting split between classes. Here we annotate source posts as *supporting*, hence we end up with a slight imbalance towards the *supporting* class. We used the ABCD dataset for experiments in Chapter 4.

3.4.2 Emergent

The Emergent data-set[5] contains rumoured claims from various rumour websites and associated news articles, collected and labelled by journalists (Ferreira and Vlachos, 2016). Each claim has a veracity label, either true, false or unverified. Each associated article is summarised into a headline and labelled to indicate whether its stance is for, against, or observing the claim, where observing indicates that the article merely repeats the claim. Hence this dataset can be used for several NLP tasks such as stance classification between claim and an article; claim verification; and focused summarisation using the article headlines. Figure 3.4 shows an example of an instance from the dataset. Topics of the claims in the dataset are world and national U.S. news as well as technology. The Emergent dataset contains 300 claims, and 2,595 associated article headlines, with an average ratio of 8.65 (7.31) articles

[5]https://github.com/willferreira/mscproject

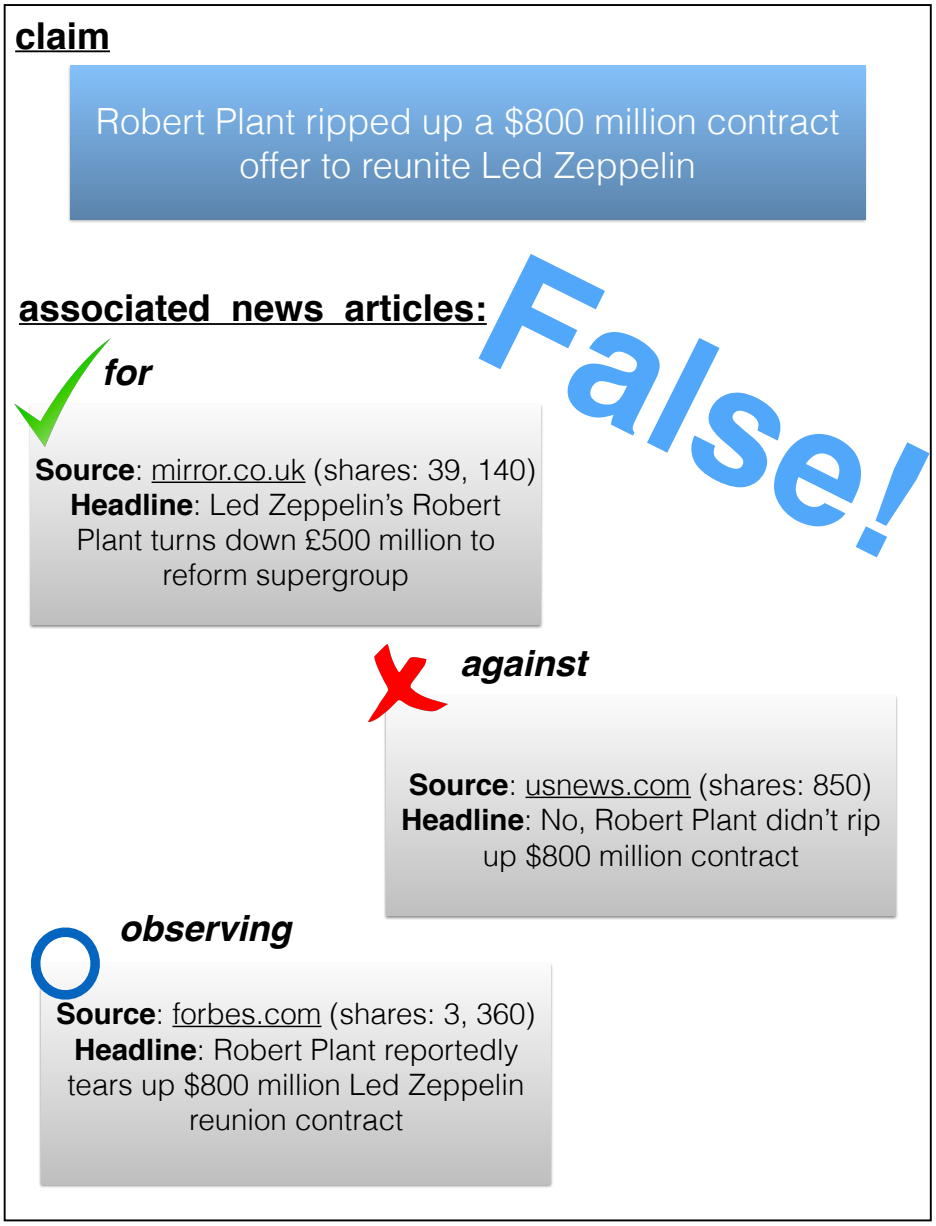


Figure 3.4: Example of a claim from the Emergent dataset.

per-claim; the minimum number of articles per-claim is 1 and the maximum number is 50. The class distribution of article stances is 47.7% for, 15.2% against and 37.1% observing. This dataset was split into training and test set parts, containing 2,071 and 524 instances respectively, ensuring that each claim appeared in only one of the parts. We used the Emergent dataset for experiments in Chapter 6.

CHAPTER 4

Sequential Approach to Rumour Stance Classification

4.1 Introduction

Stance classification is concerned with determining the attitude of the author of a text towards a target (Mohammad et al., 2016). Targets can range from abstract ideas, such as political ideologies, to concrete entities and events, for example upcoming elections. Stance classification has been studied in different domains (Ranade et al., 2013; Chuang and Hsieh, 2015), here we focus on stance classification of tweets towards the truthfulness of rumours circulating on Twitter in conversations discussing breaking news. Each conversation is defined by a tweet that initiates the conversation and a set of nested replies to it that form a conversation tree. The goal is to classify each of the tweets in the conversation tree as either *supporting*, *denying*, *querying* or *commenting* (*SDQC*) on the rumour initiated by the source tweet. Being able to detect stance automatically is very useful in the context of events provoking public resonance and associated rumours, as a first step towards verification of early reports (Zhao et al., 2015). For instance, it has been shown that rumours that are later proven to be false tend to spark significantly larger numbers of denying tweets than rumours that are later confirmed to be true (Mendoza et al., 2010; Procter et al., 2013; Zhao et al., 2015). Section 2.2.4 describes relevant works in the area of stance classification and rumour stance classification.

In this chapter we address research questions RQ3, RQ4 and RQ5 from the introduction. We focus on exploiting the structure of social media conversations for stance classification task and introduce a sequence-based neural approach to harness conversation structure (section 4.3.3). We show that approaches that utilise context in the form of conversation structure are superior to those that focus on individual replies, or reply pairs (sections 4.7.1, 4.7.2). We also perform comparison of various

feature sets that can aid classification models (section 4.7.3). Further, we present results of model performance in two shared tasks compared against performance of other participants (sections 4.7.4, 4.7.5).

We make the following contributions:

- We provide comprehensive analysis of different ways of representing conversation structure: (1) as a collection of independent tweets, (2) as a collection of pairs of tweets (each tweet in a conversation is paired with the tweet initiating the conversation), and (3) as a linear sequence of connected tweets (branch).
- We show that a sequential approach to stance classification that utilises temporal and structural information of the conversation is beneficial compared to non-sequential models.
- We perform exhaustive analysis of features relevant to the stance classification task that affect model performance and provide the resulting list of the features leading to the best performance.
- We propose *branch-LSTM* model that uses LSTM cells to process and label the linear tweet branches which outperforms non-sequential baseline approaches and all of the systems in RumourEval 2017 competition.
- We provide *branch-LSTM* as a baseline for further developments on the task of stance classification in the setting of RumourEval 2019 shared task.

4.2 Datasets

The datasets used in this chapter are the PHEME, RumourEval 2017, RumourEval 2019 and ABCD that contain stance annotations. The details on data collection process, number of instances, class balance and evaluation procedure (training/testing split or cross-validation) are provided in chapter 3. Note, only a small subset (297 trees) of the PHEME dataset is annotated with stance labels for each tweet, which covers 8 out of 9 events in the dataset. We use the PHEME dataset for model and feature selection, then use the resulting approach on the RumourEval 2017 and 2019 datasets. We use the ABCD dataset from an online debate forum to test the generalisation of our hypothesis of the importance of conversation structure and of the developed methods to a domain different than Twitter.

4.3 Methods

We compare different perspectives to deconstructing tree-like conversation structures: as single tweets, tweet pairs (source and each tweet in the conversation), and

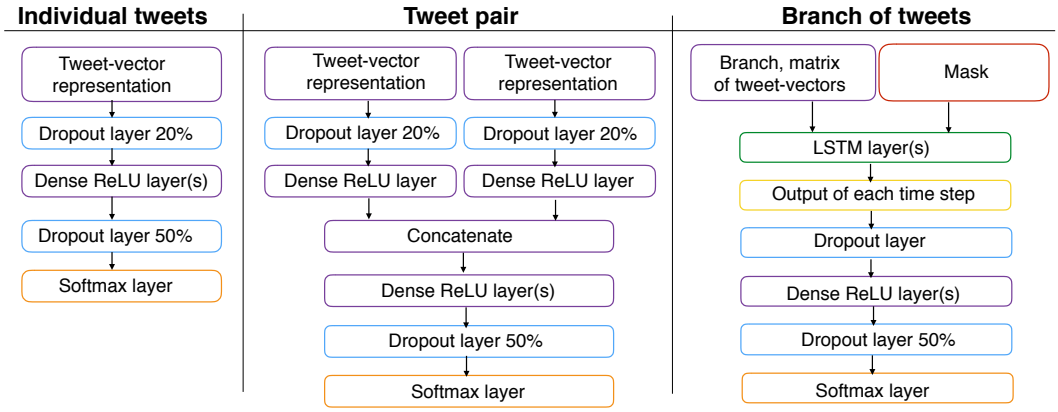


Figure 4.1: Illustration of the input/output structure of single tweet, tweet pair and *branch-LSTM* models.

tweet branches (entire linear thread). In order to do that we construct different neural network architectures for each of the three settings to classify stance of the tweets. Figure 4.1 illustrates the model architectures for these three approaches to decomposing conversation structure with corresponding models described in sections 4.3.1–4.3.3. In all of the models we use Rectified Linear Unit (ReLU) as an activation function, dropout regularisation (Srivastava et al., 2014) at the input (20%) and output (50%) levels, and a softmax layer at the end to predict the probability that an instance belongs to a certain class. All models are trained using the categorical cross entropy loss function.

4.3.1 Individual tweets

The *single-FFNN* (Feed Forward Neural Network) and *single-LSTM* are the architectures used to classify individual tweets. This approach is illustrated in figure 4.1, left. The *single-FFNN* takes tweets represented as the average of their word vectors as an input. The *single-LSTM* takes as the input a matrix of word vectors and obtains vector representations by using an LSTM layer. Then both models follow the same architecture of 20% input dropout layer, several feed-forward fully connected layers with ReLU activation functions, 50% output dropout layer and a softmax layer to provide class probabilities. The number of hidden ReLU layers λ and number of neurons ν in the layer are determined using hyper-parameter search (described in section 4.6).

To describe this model mathematically we introduce the following notation. The dataset contains K conversation trees. A single conversation k contains N_k

tweets. Thus, there are $\sum_{k=1}^{K} N_k$ tweets in the dataset. Input to the individual tweet model is a vector representation of a tweet $x_i \in \mathbb{R}^d$, where $i \in [1, \ldots, \sum_{k=1}^{K} N_k]$ and d is the number of features. Tweet x_i has true label y_i that is represented using one-hot encoding. Forward propagation of the model (left in figure 4.1) then goes as follows:

$$h_0 = x_i \odot \sigma_{in}; \quad \sigma_{in} \in \{0,1\}^d;\ (\sigma_{in})_j \sim \text{Bernoulli}(p_{in}) \tag{4.1}$$

$$h_1 = \text{ReLU}(W_1 \cdot h_0 + b_1); \quad h_0 \in \mathbb{R}^d;\ W_1 \in \mathbb{R}^{d \times \nu};\ b_1, h_1 \in \mathbb{R}^{\nu} \tag{4.2}$$

$$\ldots$$

$$h_\lambda = \text{ReLU}(W_\lambda \cdot h_{\lambda-1} + b_\lambda) \tag{4.3}$$

$$h_{\lambda+1} = h_\lambda \odot \sigma_{out}; \quad \sigma_{out} \in \{0,1\}^{\nu};\ (\sigma_{out})_j \sim \text{Bernoulli}(p_{out}) \tag{4.4}$$

$$h_{out} = \text{softmax}(W_{out} \cdot h_{\lambda+1} + b_{out}) \tag{4.5}$$

$$\text{loss} = \sum_{c=1}^{C} y_i^c \log(h_{out}^c), \tag{4.6}$$

where σ_{in}, σ_{out} are dropout masks, $p_{in} = 1 - p_{dropout} = 0.8$ is the probability of keeping a node, similarly $p_{out} = 0.5$, $\odot$ is an element-wise multiplication, ReLU is activation function as defined in equation 2.6, and $\text{softmax}(a)_i = \frac{e^{a_i}}{\sum_j e^{a_j}}$ and a_i is the ith element of the vector a, and $C = 4$ is the number of classes. Loss here is defined for an individual instance, however later it is averaged over all of the instances in the training set (or mini-batch).

4.3.2 Tweet pairs

Two models, *pair-FFNN* and *pair-LSTM* (illustrated in figure 4.1, centre), are applied to the pairs of source and response tweets and predict the label for the response tweet. As all of the tweets in the dataset are annotated, including the source tweets, and thus, to keep the number of testing instances consistent between models, we also build pairs of source tweet with itself. Pair models have two input channels that go through a 20% input dropout layer, ReLU layer and then are concatenated. After merging the resulting vector goes through several feed-forward fully connected layers with ReLU activation functions, 50% output dropout layer and a softmax layer to provide class probabilities, just as in single tweet model.

To describe this model mathematically we follow the notation introduced in section 4.3.1. Here, the model has two inputs, x_k^{src} and x_k^i, where $i \in [1, \ldots, N_k]$, and $k \in [1, \ldots, K]$ and the true stance label for each response is y_k^i. The forward propagation is then as follows:

$$h_0^{src} = x_k^{src} \odot \sigma_{in}^{src}; \tag{4.7}$$
$$h_1^{src} = \text{ReLU}(W_1^{src} \cdot h_0^{src} + b_1^{src}); \tag{4.8}$$

$$h_0^{res} = x_k^{i} \odot \sigma_{in}^{res}; \tag{4.9}$$
$$h_1^{res} = \text{ReLU}(W_1^{res} \cdot h_0^{res} + b_1^{res}); \tag{4.10}$$

$$h_{concat} = [h_1^{src}, h_1^{res}]; \tag{4.11}$$
$$h_1 = \text{ReLU}(W_1 \cdot h_{concat} + b_1); \tag{4.12}$$
$$\ldots$$
$$h_\lambda = \text{ReLU}(W_\lambda \cdot h_{\lambda-1} + b_\lambda); \tag{4.13}$$
$$h_{\lambda+1} = h_\lambda \odot \sigma_{out}; \tag{4.14}$$
$$h_{out} = \text{softmax}(W_{out} \cdot h_{\lambda+1} + b_{out}); \tag{4.15}$$
$$\text{loss} = \sum_{c=1}^{C} y_k^{ic} \log(h_{out}^c), \tag{4.16}$$

where *src* stands for source tweet of the conversation and *res* stands for response, σ_{in} and σ_{out} are dropout masks with dropout probability 0.2 and 0.5 respectively (defined as in previous section), $C = 4$ is the number of classes.

4.3.3 Branch of tweets

Here, sequential approach means incorporating the information about the whole sequence of responses (content as well as temporal and/or structural relations) as an input to the model rather than just using each post individually or in pairs. To implement a sequential approach to rumour stance classification we propose the *branch-LSTM* and *hierarchical-LSTM* architectures that use layers of LSTM units to process the whole branch of tweets, thus incorporating structural information of the conversation (see figure 4.1, right). The *branch-LSTM* model is illustrated in figure 4.3 and *hierarchical-LSTM* in figure 4.4.

To work with tweet branches they need to be extracted from the original set of conversational trees, where each tree is composed of a root node which is the source tweet and the rest are replying tweets. A tree is split into branches by creating a separate branch for each of the sequences, spanning from a leaf tweet up until the source tweet by following their parent tweets (i.e. the tweets they

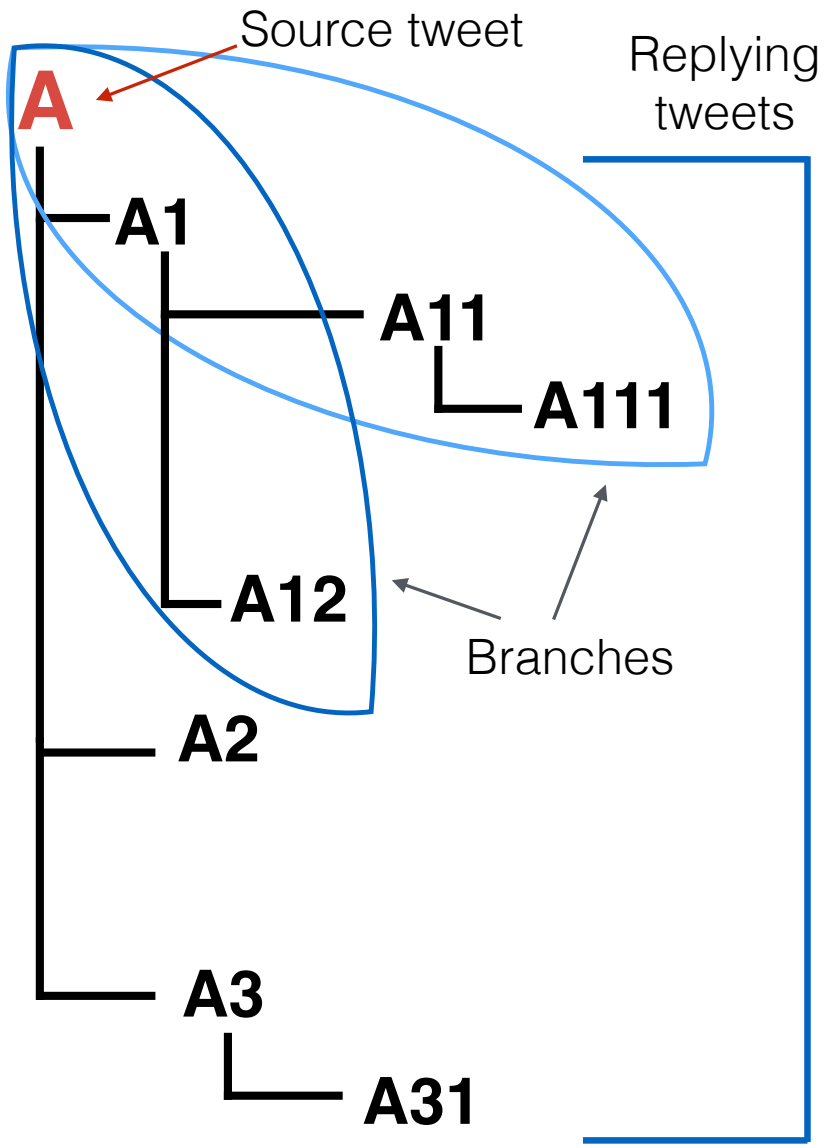


Figure 4.2: Illustration of a conversation structure and its split into branches.

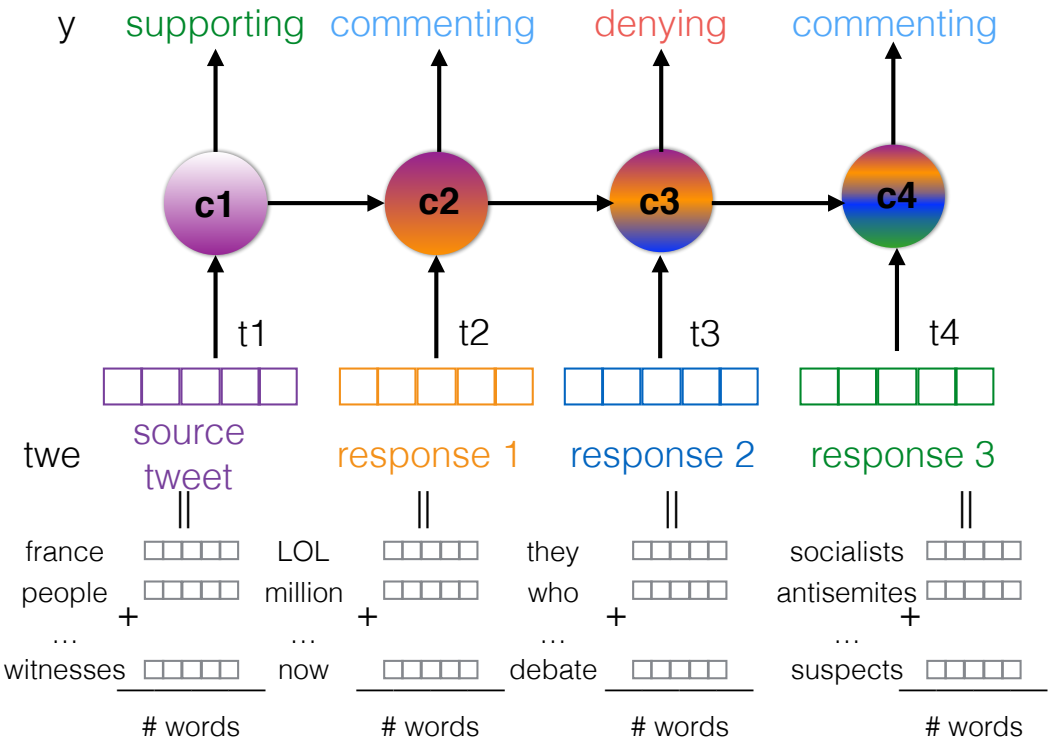


Figure 4.3: Illustration of the input/output structure of the *branch-LSTM* model.

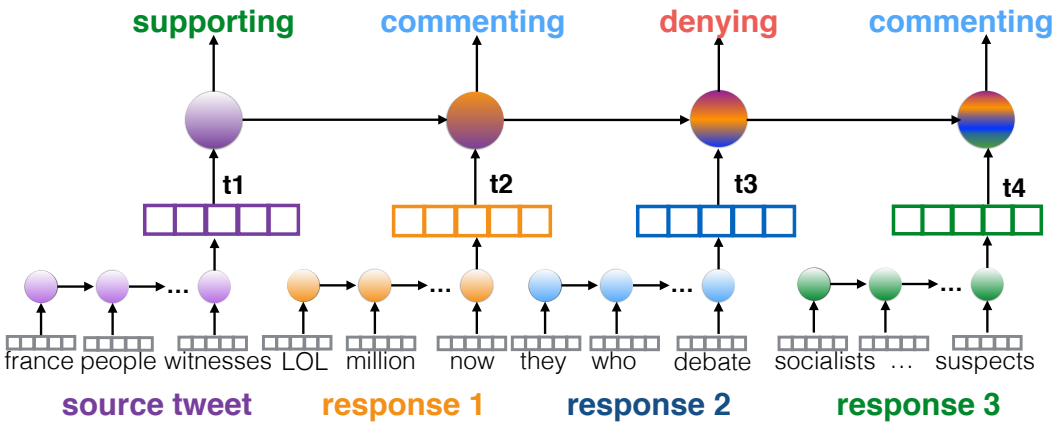


Figure 4.4: Illustration of the input/output structure of the *hierarchical-LSTM* model.

responded to). This is illustrated in figure 4.2.

Thus the input at each time step of the LSTM layer is the representation of the tweet as a vector. The *branch-LSTM* model uses the representation as an average of word vectors, and the *hierarchical-LSTM* uses a nested LSTM layer to process tweets word by word and provide the tweet representation at its final time step.

We record the output of each time step so as to attach a label to each tweet in a branch. This output is put through several dense ReLU layers (number is determined during hyper-parameter search as described below in section 4.6.3), a 50% dropout layer, and then through a softmax layer to obtain class probabilities.

We use zero-padding and masks to account for the varying lengths of tweet branches. The model is trained using the categorical cross entropy loss function and since there is overlap between branches originating from the same source tweet, we exclude the repeated tweets from the loss function using a mask at the training stage.

Again, to describe this model mathematically we follow the notation introduced two previous sections. Additionally, a conversation k can be decomposed uniquely into M_k branches (as described above). A branch is labelled z_{mk}, $m \in [1, .., M_k]$ and has a length T_{mk}. Then the forward propagation is as follows:

$$h_{0t} = \text{LSTM}(z_{mk}); \quad t \in [1, \ldots, T_{mk}] \tag{4.17}$$

$$h_{1t} = \text{ReLU}(W_1 \cdot h_{0t} + b_1); \tag{4.18}$$

$$\ldots$$

$$h_{\lambda t} = \text{ReLU}(W_{\lambda t} \cdot h_{(\lambda-1)t} + b_{\lambda t}); \tag{4.19}$$

$$h_{(\lambda+1)t} = h_{\lambda t} \odot \sigma_{out,t}; \tag{4.20}$$

$$h_{out,t} = \text{softmax}(W_{out,t} \cdot h_{(\lambda+1),t} + b_{out,t}); \tag{4.21}$$

$$\text{loss} = \sum_{c=1}^{C} y_t^c \log(h_{out,t}^c)\delta_t, \tag{4.22}$$

where the LSTM is a layer as defined in equations 2.13–2.17, here LSTM uses a sigmoid activation function, no peephole connections and returns the results of each of the time steps that are then processed to obtain per-tweet predictions, y_t is a true label for each tweet that is one-hot encoded. Variable δ_t here is a binary flag indicating whether this tweet has already been seen before by the model (value 0 if seen, 1 if not seen yet), such that the tweets overlapping between branches do not contribute to the loss multiple times.

4.3.4 Other approaches incorporating structural and/or temporal information

We also compare the above-mentioned approaches with two approaches from previous works (Zubiaga et al., 2016a; Lukasik et al., 2016) that also study and show results supporting the hypothesis that the stance classification task benefits from considering its sequential nature, thus further motivating our study.

Hawkes Processes

In work by Lukasik et al. (2016), the conversation around a rumour is modelled from the perspective of a timeline of responses. One approach for modelling the arrival of tweets around rumours is based on point processes, a probabilistic framework where tweet occurrence likelihood is modelled using an intensity function over time. Higher values of intensity function denote a higher likelihood of tweet occurrence. Thus, Lukasik et al. (2016) modelled tweet arrivals with a Hawkes Process (Hawkes, 1971) and the resulting model was applied for stance classification of tweets around rumours. Their study was only using four largest events in the PHEME dataset (Ottawa shooting, Ferguson riots, Charlie Hebdo, Sydney siege). They compare

their Hawkes Process approach, which takes into account temporal information in addition to text, with a number of baselines that only rely on the text of individual tweets, showing the importance of making use of temporal information available in tweets.

Conditional Random Fields

Work by Zubiaga et al. (2016a) also supports the hypothesis about the sequential nature of stance classification task and the importance of including the information about the conversation structure to improve model performance. They use Conditional Random Fields (CRF) to model sequences observed in Twitter conversations that allows modelling of the conversation as a graph that will be treated as a sequence of stances. Different to traditionally used classifiers for this task, which choose a label for each input unit (e.g. a tweet), CRF also consider the neighbours of each unit, learning the probabilities of transitions of label pairs to be followed by each other.

Zubiaga et al. (2016a) use all of the eight events in the PHEME dataset. They compare models that classify tweets individually (SVM, Random Forest, Naive Bayes MaxEnt) with models that process branches of tweets (Linear CRF), as well as whole conversation trees (Tree CRF). Their results show superior performance of Linear CRF and Tree CRF models over individual tweet classifiers, and also outperform the approach proposed in Lukasik et al. (2016).

4.4 Tweet representation

We experiment with different tweet representation approaches. For each of the models we perform experiments with two ways of tweet representation: (1) each tweet is encoded as the average of word2vec word embeddings (Mikolov et al., 2013a) (in *single-FFNN*, *pair-FFNN*, *branch-LSTM* models) and (2) tweet representations are created by using an LSTM layer (in *single-LSTM*, *pair-LSTM*, *hierarchical-LSTM* models). In the second approach the LSTM layer takes a word embedding as an input at each time step and the output of the final time step is taken as the tweet representation. The LSTM layer is trained together with the rest of the model. This approach is illustrated in figure 4.4 as part of *hierarchical-LSTM* model. We use two types of word embeddings: in experiments with the PHEME dataset, word vectors were created using the CBOW model (Mikolov et al., 2013a) on the complete PHEME dataset, including its unannotated part; while in experiments with the ABCD, RumourEval 2017, 2019 datasets we used publicly available word vectors

pre-trained on the Google News dataset. We did not fine-tune word embeddings while training our models.

We also perform experiments adding extra manually defined feature sets to word embedding based representations. Features used are described in section 4.5.

4.5 Relevant features

We have studied the effect of different feature types, each of which can be categorised into several subtypes of features.

4.5.1 Local Features

Local features are extracted from each of the tweets in isolation, and therefore it is not necessary to look at other features in a conversation to generate them. We use four types of features to represent the tweets locally.

Local feature type 1: Lexicon

- *Word embeddings:* we use word2vec (Mikolov et al., 2013b) to represent the textual content of each tweet. First, we trained a separate word2vec model for each of the eight folds, each having the seven events in the training set as input data, so that the event (and the vocabulary) in the test set is unknown. We use the PHEME dataset including all the tweets we collected for those events, even not annotated, as part of the data. Finally, we represent each tweet as a vector with 300 dimensions, as the average of vector representations of the words in the tweet using word2vec.

- *Part-of-speech (POS) tags:* we parse the tweets to extract the part-of-speech (POS) tags using TwitIE (Bontcheva et al., 2013). Once the tweets are parsed, we represent each tweet with a vector that counts the number of occurrences of each type of POS tag. The final vector therefore has as many features as different types of POS tags we observe in the dataset.

- *Use of negation:* this is a feature determining the number of negation words found in a tweet. The existence of negation words in a tweet is determined by looking at the presence of the following words: not, no, nobody, nothing, none, never, neither, nor, nowhere, hardly, scarcely, barely, don't, isn't, wasn't, shouldn't, wouldn't, couldn't, doesn't.

- *Use of swear words:* this is a feature determining the number of 'bad' words present in a tweet. We use a list of 458 bad words[1]. This feature is potentially indicative of a strong emotion and could be helpful for identifying stance of posts.

Local feature type 2: Content formatting

We test whether particular stance types are associated with longer posts, for example, to write explanation for one's point of view.

- *Tweet length:* the length of the tweet in number of characters.

- *Word count:* the number of words in the tweet, counted as the number of space-separated tokens.

Local feature type 3: Punctuation

- *Use of question mark:* binary feature indicating the presence or not of at least one question mark in the tweet. This feature is potentially indicative of querying posts.

- *Use of exclamation mark:* binary feature indicating the presence or not of at least one exclamation mark in the tweet. This can indicate strong sentiment and/or confidence of the aurthor of the post.

Local feature type 4: Tweet formatting

- *Attachment of URL:* binary feature, capturing the presence or not of at least one URL in the tweet. Users might attach URLs in order to prove their point of view, either supporting or denying a rumour.

4.5.2 Contextual Features

Contextual features represent the context around the tweet in question. We split them into social, structural and relational. Social features describe the interactions with other users, structural characterise the place of the tweet in the conversation structure and relational capture similarities with other tweets in the conversation.

[1] `http://urbanoalvarez.es/blog/2008/04/04/bad-words-list/`

Contextual feature type 1: Relational features

These features provide information about the parts of the conversation that are not directly included in the input to the model.

- *word2vec similarity with source tweet:* we compute the cosine similarity between the word vector representation of the current tweet and the word vector representation of the source tweet. This feature intends to capture the semantic relationship between the current tweet and the source tweet and therefore helps to infer the type of response.
- *word2vec similarity with preceding tweet:* likewise, we compute the similarity between the current tweet and the preceding tweet, i.e. the one that it is directly responding to.
- *word2vec similarity with tree:* we compute another similarity score between the current tweet and the rest of the tweets in the conversation tree excluding the tweets from the same author as that in the current tweet.

Contextual feature type 2: Structural features

- *Is leaf:* binary feature indicating if the current tweet is a leaf, i.e. the last tweet in a branch of the tree, with no more replies following. As long conversations tend to go off topic of the source tweet and last tweets are less likely to be contributing to discussion of rumour veracity.
- *Is source tweet:* binary feature determining if the tweet is a source tweet or is instead replying to someone else. Note that this feature can also be extracted from the tweet itself, checking if the tweet content begins with a Twitter user handle or not. As most of the source tweets were chosen to convey a rumour, they are often of the *supporting* stance.
- *Is source user:* binary feature indicating if the current tweet is posted by the same author as that in the source tweet. Users spreading the rumour often tend to continue supporting it throughout the conversation.

Contextual feature type 3: Social features

We analyse whether tweets of certain stance attract more reactions from the community around a rumour.

- *Has favourites:* feature indicating the number of times a tweet has been favourited.

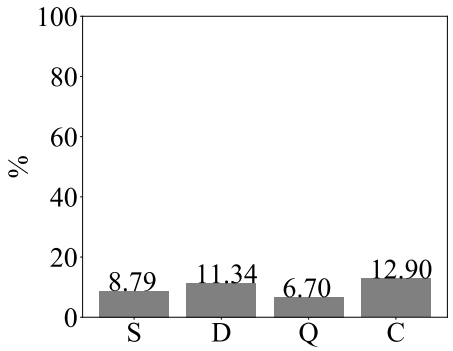
- *Has retweets:* feature indicating the number of times a tweet has been retweeted.
- *Persistence:* this feature is the count of the total number of tweets posted in the conversation tree by the author in the current tweet. High numbers of tweets in a conversation indicate that the author participates more.
- *Time difference:* this is the time elapsed, in seconds, from when the source tweet was posted to the time the current tweet was posted.

4.5.3 Features used in Lukasik et al. (2016)

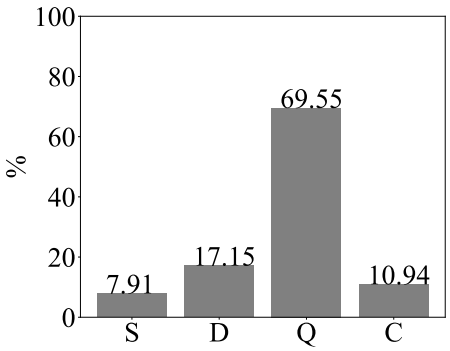
The features used in Lukasik et al. (2016), which we denote as HF, are defined as a special category because we want to perform the comparison between models while using the same set of features.

- *Bag of words:* a vector where each token in the dataset represents a feature, where each feature is assigned a number pertaining its count of occurrences in the tweet.
- *Timestamp:* The UNIX time (seconds since Jan 01 1970) in which the tweet was posted.

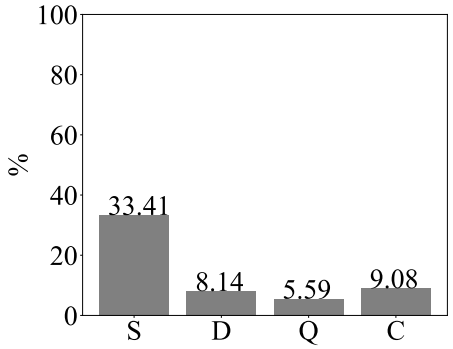
Figures 4.5 and 4.6 show distributions of selected features across each of the four stance classes. This might give an intuition of how the selected features might affect model performance prior to conducting the experiments. *Supporting* tweets tend to have URLs attached more often than other categories. Naturally, question mark most frequently indicates *querying* tweets, however it is also present in other categories. As tweets are limited in length by a platform, character count correlates strongly with word count and is in same ranges for all tweets. Negation words usage is different in *denying* tweets comparing to other categories, but it is unclear whether this distribution difference is sufficient to help identify them. Further, we perform feature selection using a wrapper method (Guyon and Elisseeff, 2003), i.e. training a new model for each subset of features and judging the usefulness of the feature set by the effect on the performance of stance classification model. Wrapper methods are computationally intensive, but usually provide the best performing feature set for the particular type of model or problem. Another way of feature selection could be the use of filter method, that scores feature sets using a proxy measure such as the mutual information (Guyon and Elisseeff, 2003), or the pointwise mutual information (Yang and Pedersen, 1997). Filter methods are generally less computationally intensive than wrappers, but the resulting feature set is not tuned to a



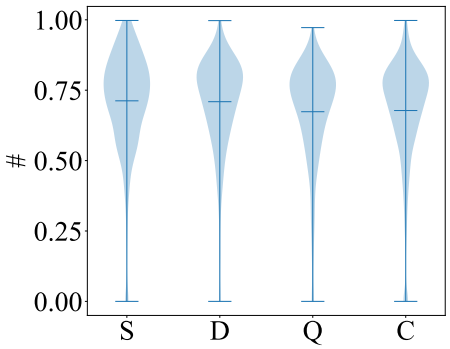
(a) Presence of exclamation mark



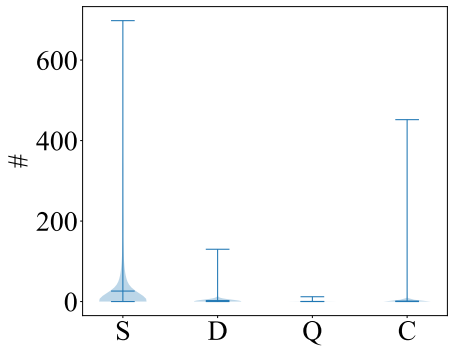
(b) Presence of question mark



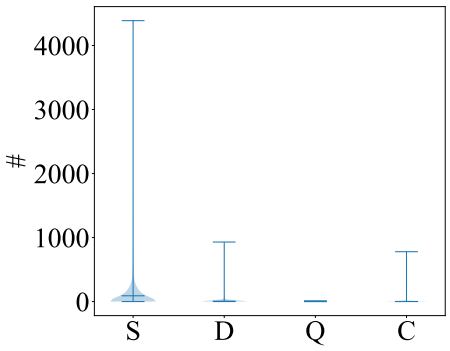
(c) Presence of URL



(d) Word2vec similarity with other tweets

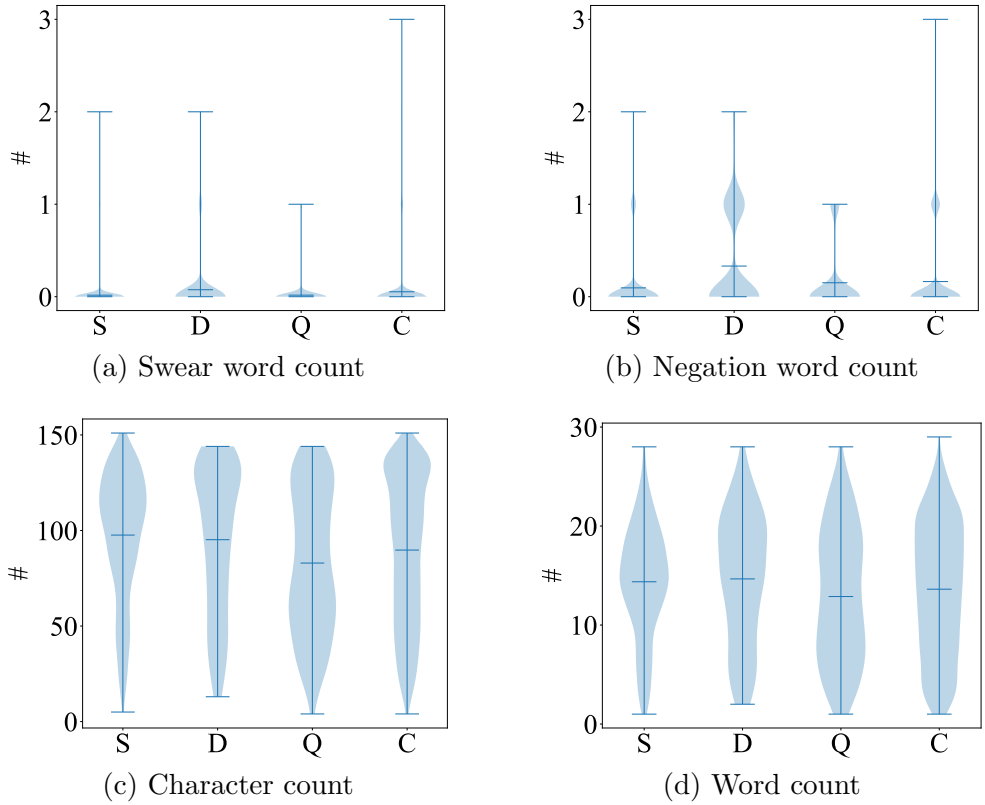


(e) Favourite count



(f) Retweet count

Figure 4.5: Distribution of binary (a-c) and count-based (d-f) features per-class in PHEME dataset (subset annotated for stance).



(a) Swear word count

(b) Negation word count

(c) Character count

(d) Word count

Figure 4.6: Distributions of count-based features per-class in PHEME dataset (subset annotated for stance).

particular type of predictive model and hence the chosen feature set would lead to lower predictive performance Zhang et al. (2013).

4.6 Experimental setup

4.6.1 Preprocessing

Prior to generating the features for the tweets, we perform a preprocessing step where we remove non-alphabetic characters, convert all words to lower case and tokenise texts.

4.6.2 Evaluation setup

For the RumourEval 2017, 2019 and ABCD datasets we use the split into training, development and testing sets provided by the authors of the dataset. Hence we train the model on training and evaluate on development set while tuning hyper-parameters, and as the parameters are chosen, we re-train the model on training plus development set to evaluate on the testing.

We split the PHEME dataset into training, development and test sets without mixing the events. The training set includes tweets for the "Charlie Hebdo", "Ferguson" and "Sydney siege" events. The test set is composed of conversations about the events "Putin missing", "Prince-Toronto", "Germanwings crash", "Ebola-Essien" and the development "Ottawa shooting". This distribution of events is based on the number of tweets in each of the events, which results in an approximate distribution of 72% for training, 17% for development and 11% for testing. We also split the PHEME dataset into separate events for 8-fold cross-validation in order to allow comparison with existing approaches. On the PHEME dataset we first fix hyper-parameters, then retrain models on the combined training and development set and finally evaluate the model on the held out test set and perform 8-fold cross-validation. It could be argued that using the Ottawa shooting event as a development set while tuning hyper-parameters may introduce bias leading to optimistic results. However, results on the fold corresponding to the Ottawa shooting does not show significant deviation from performance in other folds, so any bias due to parameter tuning on this dataset is minimal (see per-event results in table 4.4).

4.6.3 Hyper-parameters

The set of hyper-parameters includes number of layers, number of nodes in each layer, mini-batch size, number of epochs, learning rate and strength of L2-regularisation.

We determine the optimal set of hyper-parameters via testing the performance of the model on the development set for different parameter combinations. We use the Tree of Parzen Estimators (TPE) algorithm (Bergstra et al., 2011) to search the parameter space, which is defined as follows: the number of dense ReLU layers varies from one to four; the number of LSTM layers is one or two; the mini-batch size is either 32 or 64; the number of units in the ReLU layer is one of {100, 200, 300, 400, 500}, and in the LSTM layer one of {100, 200, 300}; the strength of the L2 regularisation is one of $\{0.0, 10^{-4}, 3 \cdot 10^{-4}, 10^{-3}\}$ and the number of epochs is selected from {30, 50, 70, 100}. We perform 100 trials of different parameter combinations for each of the models.

We choose the best set of hyper-parameters by assessing the minimum loss on the development set. Loss is defined as (1 – macro-averaged F score) in experiments with PHEME, ABCD and RumourEval 2019 datasets, and as (1 – accuracy score) in experiments with RumourEval 2017 dataset.

We performed preliminary experiments with replacing the unidirectional LSTMs with bidirectional LSTMs, which did not bring performance improvemnets while being more computationally intensive, thus we did not pursue it further (cross-validation scores with unidirectional LSTM: macro-F is 0.440, accuracy is 0.703, and with bidirectional LSTM: macro-F is 0.435, accuracy is 0.684).

4.6.4 Evaluation metrics

In the results section we present model performance expressed as accuracy, micro- and macro-averaged F1 score, as well as per-class F1 score and confusion matrices calculated as described in section 2.1.3. Due to the class imbalance in the PHEME, RumourEval 2017 and RumourEval 2019 datasets, we consider a macro-averaged F1 score to be a more reliable performance metric as it gives equal attention to model performance on each of the classes and allows comparison with previous work (Lukasik et al., 2016; Zubiaga et al., 2016a). However, the system submitted to the RumourEval 2017 shared task was optimised to produce the highest accuracy, as this metric was chosen as the main one by the competition organisers (Derczynski et al., 2017). In the 2019 edition of RumourEval this was noted and macro-averaged F-score was used as a main competition metric.

4.6.5 Code

We have implemented the models described above using Python 2.7 with libraries Theano (Bastien et al., 2012) and Lasagne (Dieleman et al., 2015). This imple-

mentation led to the results described in sections 4.7.1–4.7.4. Code for the *branch-LSTM* model is publicly available at the following link: `https://github.com/kochkinaelena/branchLSTM`. We have also implemented the *branch-LSTM* model using Python 3.5 and Keras library that can use Theano and Tensorflow (Abadi et al., 2015) as a backend, which is also publicly available: `https://github.com/kochkinaelena/branchLSTM/tree/Keras_branch`.

We provided a Keras implementation as a baseline for the RumourEval 2019 competition: `https://github.com/kochkinaelena/RumourEval2019`, which provided the results in section 4.7.5. We use the implementation of the Tree of Parzen Estimators (TPE) algorithm in the Hyperopt package (Bergstra et al., 2013) for hyper-parameter tuning; evaluation metrics from Scikit-learn library (Pedregosa et al., 2011); Gensim package (Řehůřek and Sojka, 2010) for handling word2vec embeddings; tokenizer and list of stop words from NLTK (Loper and Bird, 2002).

4.7 Results

In this section we first present the effects of incorporating conversation structure into models for the stance classification task, through experiments with models described in section 4.3. These represent three ways of deconstructing a conversation, into: individual posts, pairs of source with response, and linear chain branches. We explore the hypothesis that sequential information is important for the stance classification task using two datasets: PHEME and ABCD.

Further we explore the relevant features that contribute to rumour stance identification. When the best performing feature set is identified on the PHEME dataset, we evaluate it on the RumourEval 2017/2019 shared task datasets in order to scrutinise their generalisability.

4.7.1 Effect of incorporating conversation structure on PHEME dataset

Tables 4.1–4.7 present the results of our experiments on the PHEME dataset. On this dataset we have performed experiments with two types of tweet representations: (1) as an average of word2vec word embeddings (*single-FFNN*, *pair-FFNN*, *branch-LSTM*), and (2) using an LSTM layer to process word2vec word embeddings and taking the output of its last time step as our tweet representation (*single-LSTM*, *pair-LSTM*, *hierarchical-LSTM*). Word embeddings were trained on the full PHEME dataset, also including its unannotated part.

	Classifier	Features	Micro-F1	Macro-F1	S	D	Q	C
Baseline	Majority		0.64	0.19	0.00	0.00	0.00	0.78
	Linear CRF (Zubiaga et al., 2016a)		0.65	0.43	0.45	**0.11**	0.41	0.77
	Tree CRF (Zubiaga et al., 2016a)		0.66	**0.44**	0.46	0.09	**0.44**	0.77
Single	single-FFNN	word2vec	0.65	0.36	0.43	0.09	0.15	0.77
	single-LSTM	word2vec	0.58	0.36	0.41	**0.11**	0.18	0.72
	single-FFNN	word2vec + extra	**0.71**	0.41	0.50	0.00	0.33	**0.81**
	single-LSTM	word2vec + extra	0.70	0.38	0.48	0.00	0.22	**0.81**
Pair	pair-FFNN	word2vec	0.68	0.37	0.49	0.05	0.15	0.79
	pair-LSTM	word2vec	0.56	0.33	0.36	0.08	0.18	0.71
	pair-FFNN	word2vec + extra	0.70	0.32	0.45	0.00	0.01	**0.81**
	pair-LSTM	word2vec + extra	0.58	0.26	0.23	0.02	0.04	0.73
Branch	branch-LSTM	word2vec	0.65	0.39	0.48	0.07	0.22	0.78
	hierarchical-LSTM	word2vec	0.63	0.38	0.49	0.10	0.16	0.76
	branch-LSTM	word2vec + extra	0.70	**0.44**	0.51	0.00	**0.44**	**0.81**
	hierarchical-LSTM	word2vec + extra	**0.71**	**0.44**	**0.53**	0.00	0.43	**0.81**

Table 4.1: Cross-validation F1 scores on the PHEME dataset: micro- and macro-averaged, and per-class (S: *supporting*, D: *denying*, Q: *querying*, C: *commenting*).

	Classifier	Features	Micro-F1	Macro-F1	S	D	Q	C
Single	single-FFNN	word2vec	0.61	0.30	0.26	**0.18**	0.00	0.75
	single-LSTM	word2vec	0.57	0.37	0.39	0.17	0.20	0.72
	single-FFNN	word2vec + extra	0.67	0.33	0.42	0.00	0.13	**0.79**
	single-LSTM	word2vec + extra	0.66	0.37	0.40	0.00	0.31	0.77
Pair	pair-FFNN	word2vec	0.61	0.32	0.36	0.12	0.08	0.74
	pair-LSTM	word2vec	0.53	0.34	0.24	0.15	0.28	0.70
	pair-FFNN	word2vec + extra	0.65	0.28	0.33	0.00	0.00	0.78
	pair-LSTM	word2vec + extra	0.61	0.23	0.05	0.09	0.00	0.77
Branch	branch-LSTM	word2vec	0.63	0.37	0.46	**0.18**	0.08	0.75
	hierarchical-LSTM	word2vec	0.60	**0.41**	**0.51**	0.13	0.27	0.72
	branch-LSTM	word2vec + extra	0.66	**0.41**	0.43	0.00	0.44	0.77
	hierarchical-LSTM	word2vec + extra	**0.68**	**0.41**	0.35	0.00	**0.49**	**0.79**

Table 4.2: F1 scores on the testing set of the PHEME dataset: micro- and macro-averaged, and per-class (S: *supporting*, D: *denying*, Q: *querying*, C: *commenting*).

We also compare two types of features used in these experiments: (1) purely word2vec embeddings based, and (2) this word2vec representation concatenated with the following extra features: presence of a period, presence of an exclamation mark, presence of a question mark, presence of a URL, presence of images, whether the tweet has re-tweets, whether the tweet has been favourited, word count, character count and whether the tweet is the source tweet of a conversation. This set of features largely overlaps with the features used in Zubiaga et al. (2016a). In section 4.7.3 we also provide a comparison of those models with our *branch-LSTM* model, working on different sets of features, carefully choosing the best one, leading to the model with the highest performance.

The results in tables 4.1 and 4.2 show the performance of all of the model vari-

Model	True Positive	D as C	D as Q	D as S	False Positive	C as D	Q as D	S as D
word2vec features only								
single-FFMM	23	282	10	29	118	84	15	19
pair-FFMM	9	294	7	34	25	19	3	3
branch-LSTM	26	251	27	40	156	107	26	23
word2vec + extra features								
single-FFMM	0	321	8	15	0	0	0	0
pair-FFMM	0	335	0	9	0	0	0	0
branch-LSTM	0	300	18	26	0	0	0	0

Table 4.3: *Denying* tweets (D) mis-classification table for the cross-validation results on the PHEME dataset (S: *supporting*, D: *denying*, Q: *querying*, C: *commenting*).

	branch-LSTM		Tree CRF		# branches	# tweets				
Event	**Micro-F**	**Macro-F**	**Micro-F**	**Macro-F**	**Total**	**Total**	**S**	**D**	**Q**	**C**
Charlie Hebdo	**0.74**	**0.45**	0.69	0.43	776	1071	239 (22.0%)	58 (5.0%)	53 (4.0%)	721 (67.0%)
Ebola-Essien	**0.65**	**0.43**	0.63	0.38	22	34	6 (17.0%)	6 (17.0%)	1 (2.0%)	21 (61.0%)
Ferguson	**0.68**	**0.39**	0.56	**0.39**	677	1084	176 (16.0%)	91 (8.0%)	99 (9.0%)	718 (66.0%)
Germanwings	**0.70**	0.45	0.69	**0.52**	202	281	69 (24.0%)	11 (3.0%)	28 (9.0%)	173 (61.0%)
Ottawa shooting	**0.68**	**0.46**	0.6	**0.46**	494	777	161 (20.0%)	76 (9.0%)	63 (8.0%)	477 (61.0%)
Prince-Toronto	**0.73**	0.50	0.67	**0.52**	61	103	21 (20.0%)	7 (6.0%)	11 (10.0%)	64 (62.0%)
Putin missing	0.58	0.38	**0.66**	**0.45**	43	104	18 (29.0%)	6 (9.0%)	5 (8.0%)	33 (53.0%)
Sydney siege	**0.71**	0.45	0.68	**0.50**	750	1107	220 (19.0%)	89 (8.0%)	98 (8.0%)	700 (63.0%)
Total	0.70	0.44	0.66	0.44	3025	4519	910	344	358	2907

Table 4.4: Per-event results for the *branch-LSTM* model with word2vec + extra features and dataset statistics on the PHEME dataset (S: *supporting*, D: *denying*, Q: *querying*, C: *commenting*).

ations in terms of micro- and macro-averaged F1 scores, as well as macro-averaged F1 performance scores per-class evaluated using leave-one-event-out cross-validation (table 4.1) and testing set (table 4.2). The cross-validation performance is rather low as 8-fold split is not performed randomly, but by events, which have drastically different sizes that vary from 34 to 1107 tweets, and also a different class balance. Further, we mainly focus on cross-validation results, but the analysis can be repeated also purely on the testing set and leads to similar conclusions.

Due to the strong class imbalance in the dataset, a simple baseline that always opts for the majority class achieves a high micro-averaged F1 score (table 4.1), however it performs poorly if we measure macro-averaged F1 score. This highlights the importance of using macro-averaged F1 score as the key performance metric for this task.

Models that consider tweets in isolation (*single-FFNN*, *single-LSTM*) or as pairs of source and response tweets (*pair-FFNN*, *pair-LSTM*) provide noticeable improvements in terms of macro-F1 score over the majority baseline, while still showing lower performance than the result in Zubiaga et al. (2016a). When evaluating (source, response) tweet pair models we have included also (source, source) tweet pairs in the evaluation to have consistent number of testing instances between models. This led to significant improvement in classification of the *supporting* class

for these models when compared to the performance with (source, source) tweet pairs excluded, because the majority of source tweets belong to the *supporting* class.

The models that use individual tweets lack any rumour context information and models that work on tweet pairs might be missing relevant information contained in the conversation in between the source and response tweet pair. However when linear branch structure and extra features are taken into account (*branch-LSTM*, *hierarchical-LSTM*) we see significant improvements in the macro-averaged F1 score compared to the previous models and Linear CRF in Zubiaga et al. (2016a), which uses exactly the same input structure as branches.

Models that use extra features (marked as word2vec + extra in tables 4.1 and 4.2), except for *pair-LSTM*, show results outperforming the results in Zubiaga et al. (2016a) in terms of micro-average and *commenting* class F1 scores. While the models that use word2vec features only, show better results on the *denying* class; extra features are very important for identifying the *supporting* class. In particular, the binary feature indicating whether the tweet is the source of the conversation, as due to the rules of construction of this dataset most (but not all) of the source tweets are conveying a rumour, hence supporting it.

The models that use LSTM layers to create tweet vector representations do not improve macro-F1 score when compared to those that represent tweets as an average of word vectors. This could be explained by the short length of the tweets and the relatively small size of the dataset. However since this model is the most computationally expensive, we only tried a few hyper-parameter combinations comparing to other models, and further exploration of the parameter space could lead to further improvements.

Table 4.3 shows the numbers of mis-classified tweets focusing on the *denying* class, the minority class, which is the hardest one to identify correctly. *Denying* and *questioning* classes are both minority classes and are very important to detect in order to evaluate public opinion on the circulating rumour.

The distribution of incorrectly classified *denying* in cases using word2vec features shows equally many false positive and false negative predictions, however the model with extra features does not predict any instances as *denying*. In the models using word2vec features only there are a lot of instances that belong to other classes, but are falsely identified as *denying*, however when extra features are used the models do not classify any other class as *denying*. Most *denying* tweets get mis-classified as *commenting*, the majority class. When we tried performing weight adjustment to make up for the class imbalance, it improved the classification of *denying* instances at the cost of overall performance.

Depth Group	# Posts	# S	# D	# Q	# C	Macro-F	Micro-F	S	D	Q	C
0	297	277	9	0	11	0.254	0.609	0.757	0.143	0	0.114
1	2602	485	213	229	1675	0.348	0.621	0.337	0.114	0.184	0.757
2	553	66	34	45	408	0.310	0.705	0.180	0.105	0.118	0.835
3	313	18	26	22	247	0.245	0.712	0	0	0.143	0.835
4	195	14	16	2	144	0.287	0.697	0	0.240	0.087	0.822
5	129	14	7	10	98	0.247	0.674	0.174	0	0	0.813
6+	430	36	39	31	324	0.233	0.691	0.109	0	0	0.823

Table 4.5: Breakdown of results of *single-FFNN* model on PHEME dataset by different depths of posts in the conversation.

Depth Group	# Posts	# S	# D	# Q	# C	Macro F	Micro F	S	D	Q	C
0	297	277	9	0	11	0.290	0.724	0.840	0.138	0	0.182
1	2602	485	213	229	1675	0.342	0.631	0.375	0.042	0.186	0.762
2	553	66	34	45	408	0.257	0.709	0.069	0.053	0.077	0.830
3	313	18	26	22	247	0.255	0.760	0	0.069	0.087	0.864
4	195	14	16	2	144	0.289	0.733	0.111	0.118	0.087	0.843
5	129	14	7	10	98	0.244	0.744	0.125	0	0	0.852
6+	430	36	39	31	324	0.213	0.744	0	0	0	0.853

Table 4.6: Breakdown of results of *pair-FFNN* model on PHEME dataset by different depths of posts in the conversation.

Depth Group	# Posts	# S	# D	# Q	# C	Macro-F	Micro-F	S	D	Q	C
0	297	277	9	0	11	0.322	0.933	0.965	0	0	0
1	2602	485	213	229	1675	0.319	0.587	0.206	0.059	0.272	0.741
2	553	66	34	45	408	0.317	0.689	0.16	0.115	0.167	0.825
3	313	18	26	22	247	0.291	0.706	0.118	0.222	0	0.824
4	195	14	16	2	144	0.283	0.708	0	0.105	0.188	0.84
5	129	14	7	10	98	0.242	0.667	0.166	0	0	0.804
6+	430	36	39	31	324	0.241	0.709	0.080	0	0.054	0.831

Table 4.7: Breakdown of results of *branch-LSTM* model on PHEME dataset by different depths of posts in the conversation.

We also perform fine-grained per-event comparison with Zubiaga et al. (2016a) (as can be seen in table 4.4). The *branch-LSTM* model outperforms Tree CRF in terms of micro-F1 scores for all the events except Putin missing, and also in terms of macro-F1 for half of the events, including the two largest events, namely Charlie Hebdo and Ferguson.

Tables 4.5–4.7 show per-depth performance breakdown of *single-FFNN*, *pair-FFNN* and *branch-LSTM* models on PHEME dataset. In those tables we also see per-depth distribution of each of the classes. Depth is defined as number of tweet in the branch starting from 0 being source tweet. Tables show that most of the tweets are concentrated at depths 2 and 3. While *commenting* is a majority class in the full dataset and depths 1-6+, most of the source tweets are *supporting*, i.e. conveying the rumour. Hence we can see that *branch-LSTM* and *pair-FFNN* models that are able to identify source tweets easily perform better on this class at depth

0 than single tweet model. In all of the models we do not observe a strong trend of increasing or decreasing accuracy or macro-F score with the depth of the tweet, this could be due to concentration of majority of tweets at depth 2. Also, we did not find notable difference in length of the tweet (number of words) between correctly and mis-classified examples for all models.

These results show that models that are processing sequential information as input for stance classification consistently outperform those which do not include information about the conversational context. Therefore they support the hypothesis of the sequential nature of the rumour stance classification task, inline with the outcomes of previous research (Zubiaga et al., 2016a; Lukasik et al., 2016).

4.7.2 Effect of incorporating conversation structure on ABCD dataset

We test our claim about the sequential nature of the stance classification task on the ABCD dataset. The ABCD dataset is described in section 3.4.1, which details the instances in the dataset, and how the task and labels in the dataset were adjusted to match the setup of stance classification in the PHEME dataset. We are testing the applicability of the sequential approach to stance classification on the debate domain, rather than discussions of rumours. This is a new setting with different types of posts, which are unlimited in length unlike tweets.

We perform experiments on the ABCD dataset training on its training set, choosing hyper-parameters using the development set and evaluating model performance on the testing set. We use word2vec features only, with the word2vec model pre-trained on the Google News dataset (300d) (Mikolov et al., 2013a), and with posts represented as an average of word vectors (*single-FFNN*, *pair-FFNN*, *branch-LSTM*).

Table 4.8 shows the results of experiments on the ABCD dataset. Performance on this dataset increases as we increase the complexity of the structure used, from single posts through pairs to linear branches. The *branch-LSTM* model shows the best performance in terms of accuracy and macro-F score. The *single-FFNN* model is not able to deal with the class imbalance and shows a poor F score for the *denying* class.

Tables 4.9–4.11 demonstrate how the *single-FFNN*, *pair-FFNN* and *branch-LSTM* models perform on posts with different depth in the branch, where depth is the number of direct parents of the post starting from the root node. *Branch-LSTM* and *pair-FFNN* models are able to classify all depth 0 posts correctly as these are only *supporting* posts, whereas the single post model *single-FFNN* is unable to detect this aspect of the dataset. This artefact boosts model performance and

	Accuracy	**Macro-F**	**Supporting**	**Denying**
Single-FFNN	0.51	0.60	0.72	0.29
Pair-FFNN	0.64	0.61	0.70	0.52
Branch-LSTM	**0.71**	**0.70**	**0.76**	**0.64**

Table 4.8: Results of three approaches to decomposing conversation structure on the ABCD dataset.

Depth Group	**# Posts**	**# Denying**	**Macro-F**	**Micro-F**	**Supporting**	**Denying**
0	3554	0	0.488	0.954	0.976	0
1	4661	2756	0.431	0.476	0.591	0.271
2	2754	753	0.473	0.663	0.790	0.157
3	1978	1330	0.410	0.422	0.494	0.326
4	1255	376	0.468	0.614	0.747	0.188
5	881	601	0.414	0.419	0.467	0.360
6+	3539	1740	0.490	0.514	0.601	0.379

Table 4.9: Breakdown of results of *single-FFNN* model on ABCD test set by different depths of posts in the conversation.

Depth Group	**# Posts**	**# Denying**	**Macro-F**	**Micro-F**	**Supporting**	**Denying**
0	3554	0	1	1	1	0
1	4661	2756	0.595	0.595	0.594	0.595
2	2754	753	0.484	0.539	0.653	0.316
3	1978	1330	0.569	0.578	0.508	0.631
4	1255	376	0.464	0.506	0.616	0.311
5	881	601	0.551	0.562	0.479	0.624
6+	3539	1740	0.501	0.501	0.513	0.489

Table 4.10: Breakdown of results of *pair-FFNN* model on ABCD test set by different depths of posts in the conversation.

Depth Group	**# Posts**	**# Denying**	**Macro-F**	**Micro-F**	**Supporting**	**Denying**
0	3554	0	1	1	1	0
1	4661	2756	0.600	0.634	0.484	0.716
2	2754	753	0.502	0.654	0.777	0.226
3	1978	1330	0.580	0.659	0.398	0.762
4	1255	376	0.486	0.678	0.800	0.172
5	881	601	0.548	0.657	0.326	0.770
6+	3539	1740	0.615	0.617	0.639	0.591

Table 4.11: Breakdown of results of *branch-LSTM* model on ABCD test set by different depths of posts in the conversation.

could have been removed or replaced with a rule. However we kept those instances in order to keep similarity with experiments on PHEME and RumourEval datasets and, also, to demonstrate that single-post model is unable to capture this artefact without the use of additional feature indicating whether the tweet is a source of a conversation. We have also noticed that longer posts get mis-classified significantly more often than shorter ones, which is due to the simplistic representation of posts as the average of word vectors.

Overall, this set of experiments confirms our hypothesis about the sequential nature of the stance classification task and the importance of the use of context.

4.7.3 Effect of adding relevant features on PHEME dataset

In this section we evaluate the performance of models processing sequential information (LSTM, Linear and Tree CRF (Zubiaga et al., 2016a), Hawkes processes (Lukasik et al., 2016)) and baseline (single tweet) models (NB, SVM, Random Forest, MaxEnt) with different feature sets.

Local features

We first look at the effect of using different combinations of local features. Table 4.12 provides a summary of the feature categories and the short names in brackets used in the result tables. Based on the results presented in table 4.13 we make the following conclusions.

- *branch-LSTM* consistently performs very well with different features in terms of both micro- and macro-F1 scores.
- Classifiers processing sequences of responses show superior performance to the non-sequential classifiers in terms of macro-F1 scores. While the two CRF alternatives perform very well, the *branch-LSTM* classifier is slightly superior. Moreover, the CRF classifiers outperform their non-sequential counterpart MaxEnt, which achieves an overall lower performance.
- The *branch-LSTM* classifier is in fact superior to the Tree CRF classifier. While the Tree CRF needs to make use of the entire tree for the classification, the *branch-LSTM* classifier only uses branches, reducing the amount of data and complexity that needs to be processed in each sequence.
- Local features that use word2vec-based tweet representation perform better than the features from Lukasik et al. (2016) that use BOW text representa-

Local features	
Lexicon (LF1)	Word embeddings POS tags Negation Swear words
Content formatting (LF2)	Tweet length Word count
Punctuation (LF3)	Question mark Exclamation mark
Tweet formatting (LF4)	URL attached
Contextual features	
Relational (R)	Similarity with source tweet Similarity with preceding tweet Similarity with tree
Structural (ST)	Is leaf Is source tweet Is source user
Social (SO)	Has favourites Has retweets Persistence Time difference
Features from Lukasik et al. (2016)	
HF	Bag of words Timestamp

Table 4.12: List of features and short names for result tables.

tions. HF achieve high performance in terms of micro-F1 scores, but leave much to be desired when we look at macro-F1 scores.

- Among the local features, combinations of subgroups of features lead to clear improvements with respect to single subgroups without combinations in terms of macro-F1.

- Even though the combination of all local features achieves good performance, there are alternative leave-one-out combinations that perform better. This combination is different depending on whether we want to achieve high micro-F1 or macro-F1 scores. However, the feature combination leading to the best balance of micro- and macro-F1 scores is that combining lexicon, content formatting and punctuation (i.e. LF123). While it is the best in terms of macro-F1 score (0.449), it is only slightly behind (0.703) the best combination in terms of micro-F1 (0.707).

Micro-F1										
	HF	LF1	LF2	LF3	LF4	LF123	LF124	LF134	LF234	LF1234
NB	0.272	0.158	0.445	0.305	0.657	0.183	0.185	0.165	0.513	0.188
SVM	0.627	0.554	0.362	0.299	0.657	0.602	0.597	0.586	0.637	0.602
Random Forest	0.682	0.665	0.414	0.297	0.645	0.665	0.667	0.671	0.520	0.668
MaxEnt	0.532	0.556	0.560	0.303	0.647	0.599	0.568	0.596	0.642	0.605
Hawkes-approx	0.625	–	–	–	–	–	–	–	–	–
Hawkes-grad	0.491	–	–	–	–	–	–	–	–	–
Linear CRF	0.610	0.552	0.540	0.702	0.696	0.546	0.512	0.584	0.613	0.547
Tree CRF	0.644	0.546	0.576	0.315	0.308	0.585	0.556	0.565	0.632	0.584
branch-LSTM	0.702	0.583	0.702	0.701	0.703	0.703	0.685	0.578	**0.707**	0.668
Macro-F1										
	HF	LF1	LF2	LF3	LF4	LF123	LF124	LF134	LF234	LF1234
NB	0.157	0.183	0.231	0.265	0.433	0.208	0.212	0.191	0.375	0.214
SVM	0.336	0.356	0.231	0.258	0.313	0.403	0.365	0.403	0.420	0.408
Random Forest	0.325	0.308	0.276	0.267	0.437	0.322	0.310	0.351	0.357	0.329
MaxEnt	0.338	0.363	0.272	0.263	0.428	0.415	0.363	0.421	0.427	0.422
Hawkes-approx	0.309	–	–	–	–	–	–	–	–	–
Hawkes-grad	0.307	–	–	–	–	–	–	–	–	–
Linear CRF	0.362	0.357	0.268	0.318	0.317	0.413	0.365	0.403	0.425	0.412
Tree CRF	0.350	0.375	0.285	0.221	0.217	0.433	0.385	0.413	0.436	0.433
branch-LSTM	0.318	0.362	0.318	0.407	0.419	**0.449**	0.395	0.412	0.429	0.437

Table 4.13: Micro- and macro-F1 performance results using local features. HF: features from Lukasik et al. (2016). LF: local features, where numbers indicate subgroups of features as follows, 1: Lexicon, 2: Content formatting, 3: Punctuation, 4: Tweet formatting.

Micro-F1									
LF=LF123	LF	R	ST	SO	LF+R+ST	LF+R+SO	LF+ST+SO	R+ST+SO	All
NB	0.183	0.649	0.145	0.162	0.176	0.163	0.167	0.175	0.167
SVM	0.602	0.682	0.702	0.336	0.633	0.466	0.607	0.536	0.568
Random Forest	0.665	0.654	0.312	0.564	0.698	0.695	0.703	0.675	0.695
MaxEnt	0.599	0.679	0.702	0.696	0.620	0.634	0.639	0.699	0.636
Linear CRF	0.546	0.702	0.702	0.671	0.603	0.632	0.631	0.672	0.635
Tree CRF	0.585	0.677	0.702	0.662	0.593	0.660	0.661	0.429	0.663
branch-LSTM	**0.703**	0.701	0.702	0.697	0.688	0.680	0.672	0.698	0.667
Macro-F1									
	LF	R	ST	SO	LF+R+ST	LF+R+SO	LF+ST+SO	R+ST+SO	All
NB	0.208	0.319	0.157	0.170	0.203	0.173	0.178	0.186	0.178
SVM	0.403	0.335	0.318	0.260	0.429	0.347	0.388	0.295	0.375
Random Forest	0.322	0.325	0.269	0.328	0.356	0.358	0.376	0.343	0.364
MaxEnt	0.415	0.333	0.318	0.310	0.434	0.447	0.447	0.318	0.449
Linear CRF	0.413	0.318	0.318	0.334	0.424	0.431	0.431	0.342	0.437
Tree CRF	0.433	0.322	0.317	0.312	0.425	0.429	0.430	0.232	0.433
branch-LSTM	**0.449**	0.318	0.318	0.315	0.445	0.436	0.448	0.314	0.437

Table 4.14: Micro- and macro-F1 performance results incorporating contextual features with local features LF=LF123, R: relational features, ST: structural features, SO: social features.

Micro-F1								
LF=LF1234	LF	LF+R	LF+ST	LF+SO	LF+R+ST	LF+R+SO	LF+ST+SO	All
NB	0.188	0.190	0.196	0.165	0.196	0.165	0.175	0.175
SVM	0.594	0.592	0.603	0.552	0.603	0.552	0.554	0.553
Random Forest	0.668	0.689	0.694	0.690	0.697	0.693	0.703	0.690
MaxEnt	0.605	0.601	0.624	0.642	0.621	0.642	0.638	0.635
Linear CRF	0.547	0.539	0.560	0.636	0.557	0.638	0.639	0.640
Tree CRF	0.584	0.583	0.598	0.664	0.596	0.665	0.668	0.668
branch-LSTM	0.668	0.692	0.696	0.662	**0.703**	0.678	0.679	0.676
Macro-F1								
	LF	LF+R	LF+ST	LF+SO	LF+R+ST	LF+R+SO	LF+ST+SO	All
NB	0.214	0.217	0.225	0.174	0.226	0.174	0.186	0.186
SVM	0.424	0.422	0.417	0.301	0.417	0.302	0.303	0.303
Random Forest	0.329	0.366	0.350	0.356	0.360	0.363	0.369	0.360
MaxEnt	0.422	0.423	0.436	0.456	0.438	0.451	0.448	0.447
Linear CRF	0.412	0.412	0.422	0.438	0.420	0.441	0.441	0.441
Tree CRF	0.433	0.434	0.446	0.428	0.445	0.428	0.431	0.435
branch-LSTM	0.437	**0.469**	0.449	0.442	0.448	0.446	0.451	0.445

Table 4.15: Micro- and macro-F1 performance results incorporating contextual features with local features LF=LF1234, R: relational features, ST: structural features, SO: social features.

Contextual features

Since we have now made choice of the best subset of local features, we explore the effect of contextual features on model performance. We perform two sets of experiments: (1) with the above selected best local feature combination (LF123) (table 4.14) and (2) all local features (LF1234) (table 4.15), because the effect on the model can also be derived from interactions between the features.

Tables 4.14 and 4.15 show results for the classifiers incorporating contextual features along with local features. We make the following observations from these results:

- The use of contextual features leads to substantial improvements for non-sequential classifiers, getting closer to and in some cases even outperforming some of the classifiers processing sequential inputs.

- Classifiers processing sequences of responses, however, do not benefit much from using contextual features. It is important to note that classifiers processing sequences of responses take the surrounding context into consideration when they aggregate sequences in the classification process. This shows that the inclusion of contextual features is not needed for classifiers representing sequential approach, given that they implicitly include context through the use of sequences.

- In fact, for *branch-LSTM* with LF123, which is still the best-performing classifier among those using local feature set LF123, it is better to only rely on local features, as the rest of the features do not lead to any improvements.

- Non-sequential classifiers that incorporate contextual features improve substantially in performance, achieving similar macro-averaged scores in some cases (e.g. MaxEnt classifier with All features versus *branch-LSTM* with LF features), however with lower micro-averaged scores. This reinforces the importance of incorporating context in the classification process, which leads to improvements for the non-sequential classifier when contextual features are added, but especially for classifiers processing sequences of responses that can natively handle context.

- The best performing model is *branch-LSTM* with a feature combination of local feature set LF1234 and relational features. Relational features add further information about the conversation tree outside the branch to the model.

4.7.4 Results on RumourEval 2017 dataset

Informed by previous experiments we chose *branch-LSTM* as the model to submit to the RumourEval 2017 stance classification task. The feature set was selected on the basis of results described in the previous section and further tuned on the development set of RumourEval 2017 data. Each tweet is represented as an average of word2vec word vectors (Mikolov et al., 2013a) pre-trained on the Google News dataset (300d)[2]. Then, it is concatenated with the following extra features: (1) count of negation words, (2) count of swear words, (3) presence of a period, (4) presence of an exclamation mark, (5) presence of a question mark, (6) ratio of capital letters, (7) presence of a URL, (8) presence of images, (9) word2vec cosine similarity with source tweet, (10) word2vec cosine similarity with preceding tweet, (11) word2vec cosine similarity with tree, (12) word count, (13) character count and (14) whether the tweet is the source tweet of a conversation.

The performance of our *branch-LSTM* model on the testing and development set of the RumourEval 2017 dataset is shown in table 4.17. Together with the accuracy, we show macro-averaged F score and per-class F-scores, as these metrics account for the class imbalance. The difference in accuracy between testing and development sets is minimal, however we see a significant difference in macro-F

[2]We have also tried using Glove word embeddings trained on a Twitter dataset (Pennington et al., 2014), but it led to a decrease in performance on both development and testing sets when compared to the approach using Google News word2vec word vectors.

Rank	System	Macro-F score
1	**Turing (branch-LSTM)**	0.784
2	Uwaterloo	0.780
3	ECNU	0.778
4	Mama Edha	0.749
5	NileTMRG	0.709

Table 4.16: Top of the leaderboard of the RumourEval 2017 competition.

	Accuracy	Macro-F	S	D	Q	C
Development	0.782	0.561	0.621	0.000	0.762	0.860
Testing	**0.784**	0.434	0.403	0.000	0.462	0.873

Table 4.17: Results of the *branch-LSTM* model for stance classification on the development and testing sets of the RumourEval 2017 dataset. Accuracy and F1 scores: macro-averaged and per-class (S: *supporting*, D: *denying*, Q: *querying*, C: *commenting*).

score due to a different class balance in these sets. The macro-F score could be improved if we used it as a metric when optimising hyper-parameters.

Table 4.16 shows the top of leaderboard in the shared task. Our method outperforms all other systems submitted to the competition and sets the state-of-the-art for rumour stance classification, being the only method that utilised conversation structure.

The *branch-LSTM* model predicts *commenting*, the majority class, well however it is unable to pick out any *denying*, which is the most challenging under-represented class. Most *denying* instances get misclassified as *commenting* (see table 4.18), with only one tweet misclassified as *querying* and two as *supporting* (figure 4.7). The problem of class imbalance in this dataset, and the task in general, can be addressed using over- or under-sampling, cost-sensitive learning or addition of extra labelled data from minority classes.

As we were considering conversation branches, it is interesting to analyse the performance distribution across different tweet depths as shown in table 4.19. The maximum branch length in the testing set is 13 with most tweets concentrated at depths from 0 to 3. Source tweets (depth zero) are predominantly *supporting* and the model predicts these very well, but performance of *supporting* tweets at other depths decreases. The model does not show a noticeable difference in performance on tweets of varying lengths (for both number of words and character count).

Label \ Prediction	C	D	Q	S
Commenting	760	0	12	6
Denying	68	0	1	2
Querying	69	0	36	1
Supporting	67	0	1	26

Table 4.18: Confusion matrix for testing set predictions.

[As querying] @username Weren't you the one who abused her?

[As supporting] "Go online & put down 'Hillary Clinton illness,'" Rudy says. Yes – but look up the truth – not health smears https://t.co/EprqiZhAxM

[As supporting] @username I demand you retract the lie that people in #Ferguson were shouting "kill the police", local reporting has refuted your ugly racism

[As commenting] @FoxNews six years ago... real good evidence. Not!

Figure 4.7: Examples of misclassified denying tweets.

4.7.5 Results on RumourEval 2019 dataset

After the success of our *branch-LSTM* system on the RumourEval 2017 shared task, we have provided this system as baseline for the 2019 edition of the task. Unlike the RumourEval 2017 task, RumourEval 2019 used macro-averaged F score as a competition metric in order to take the aforementioned class imbalance into account.

The *branch-LSTM* baseline reached an accuracy of 0.841 with macro-averaged F score of 0.493 on the testing set of the RumourEval 2019 dataset. The top scores are shown in the table 4.20, with the full leaderboard presented in Gorrell et al. (2019). Out of 22 submitted systems almost all of them have outperformed the majority baseline, while *branch-LSTM* was outperformed by 3 systems, namely BLCU

Depth	# tweets	# S	# D	# Q	# C	Accuracy	Macro-F	S	D	Q	C
0	28	26	2	0	0	0.929	0.481	**0.963**	0.000	0.000	0.000
1	704	61	60	81	502	0.739	0.348	0.000	0.000	**0.550**	**0.842**
2	128	3	6	7	112	0.875	0.233	0.000	0.000	0.000	**0.933**
3	60	2	1	5	52	0.867	0.232	0.000	0.000	0.000	**0.929**
4	41	0	0	3	38	0.927	0.481	0.000	0.000	0.000	**0.962**
5	27	1	0	1	25	0.926	0.321	0.000	0.000	0.000	**0.961**
6+	61	1	2	9	49	0.803	0.223	0.000	0.000	0.000	**0.891**

Table 4.19: Number of tweets per-depth and performance at each level of depth.

Rank	System	Macro-F score
1	BLCU NLP	0.6187
2	BUT-FIT	0.6067
3	eventAI	0.5776
*	**branch-LSTM**	0.4929
4	UPV-28-UNITO	0.4895
*	Majority baseline	0.2234

Table 4.20: Top of the leaderboard of the RumourEval 2019 competition.

	Accuracy	Macro-F	S	D	Q	C
Twitter	0.778	0.486	0.478	0.08	0.519	0.869
Reddit	0.929	0.440	0.188	0	0.607	0.967
Total	0.841	0.493	0.437	0.071	0.550	0.913

Table 4.21: Result of *branch-LSTM* model on testing set of RumourEval 2019, per-class and per-platform in terms of accuracy and macro-averaged F score.

NLP, BUT-FIT and eventAI.

The best performing system in task A (BLCU-NLP) (Yang et al., 2019) use pre-trained contextual embedding representations OpenAI GPT (Radford et al., 2018). While most systems use single tweets or pairs of tweets (source-response) as their underlying structure to operate on, BLCU-NLP employ an inference chain-based system for this paper. Specifically they consider the conversation starting with a source tweet, followed by replies, in which each one responds to an earlier one in time sequence. They take each conversation as an inference chain and concentrate on utilizing it to solve the class imbalance problem. They also have augmented the training data with external public datasets. Judging from the approaches of the best performing systems one could infer that, for subtask A, considering the sequence of earlier posts is important to correctly identify the stance of a post towards the rumour.

Table 4.21 shows overall and per-class performance of *branch-LSTM* baseline

Label \ Prediction	S	D	Q	C
Supporting	51	3	2	101
Denying	1	4	1	95
Querying	9	1	44	39
Commenting	15	4	20	1437

Table 4.22: Confusion matrix for RumourEval 2019 testing set predictions.

on the testing set of RumourEval 2019 for both the Twitter and Reddit parts of the set. Performance in terms of macro-F score is somewhat worse on the Reddit subset with no correctly identified *denying* posts. The model performs worse on supporting and denying Reddit posts than Twitter posts. Table 4.22 shows the confusion matrix of the *branch-LSTM* baseline on the testing set of RumourEval 2019. As was the case with RumourEval 2017 and PHEME, due to the class imbalance, the model predicts the majority *commenting* class very well and performs rather poorly on *denying* posts, most of which are mis-classified as *commenting*. *Supporting* posts are also more often mis-classified as *commenting* than predicted correctly. As the *questioning* class has a very easy indicator, a question mark, which is also used by human annotators to spot questions, it is mostly classified correctly.

4.8 Conclusions

In this chapter we have studied approaches to automated stance identification in conversations as a 4-way classification task with focus on Twitter conversations discussing rumours. Our first research goal was to test the hypothesis of the sequential nature of the task, i.e. the importance of conversation context for identifying the stance of an individual post. Hence, we performed a comparison of classification models that process posts individually with those working with post pairs, branches of posts and the whole conversation trees. Models, taking into account branches and trees of posts were shown to outperform models working on single and pairs of posts.

Further, we selected a suitable tweet representation and evaluated the effect of adding manually selected extra features representing various aspects of the data – local, relational, structural and social – on model performance. As a result we identified a specific set of important features, that lead to improved model performance due to their relevance to the task as well as the properties of the dataset.

The resulting model won the RumourEval 2017 competition setting a state-of-the-art at the time, however analysing the model predictions it is clear that there is still scope for improvement. Identifying *denying* posts is a challenge as it is a minority class in all datasets used and, as no features directly indicate the *denying* class, this is also the class with the lowest inter-annotator agreement (Zubiaga et al., 2016b). Therefore taking measures to overcome that is important for rumour stance classification systems, such as defining suitable metrics, adjusting loss weights, under or over-sampling.

While the Tree CRF model outperformed Linear CRF (Zubiaga et al., 2016a),

a neural approach with *branch-LSTM* was shown to perform better, while processing the same input structure as Linear CRF. This leads us to believe that there is potential for further improvement in developing neural and/or potentially a hybrid neural CRF model stance classification models that take the whole tree structures as an input.

We have tested our selected model in a competition setting against systems by other participants. Models that have outperformed *branch-LSTM* have used the more recent language representation models, namely Open AI GPT (Radford et al., 2018), ELMO (Peters et al., 2018) and BERT (Devlin et al., 2018) in their approaches, with the best performing one (BLCU NLP) also taking advantage of conversation links between the posts.

CHAPTER 5

Exploring Sequential Approach to Rumour Verification

5.1 Introduction

In this chapter we present our experiments studying rumour verification as a supervised learning problem. Rumour verification is a 3-way classification task, where a machine learning model is trained to predict whether a given rumour is *true*, *false*, or remains *unverified* (as defined in section 2.2.1). This is a data-driven approach to find patterns, which can be linguistic or social media platform-specific, that are indicative of the true or false nature of a rumour.

We study two specific research questions (RQ1 and RQ2 from chapter 1). Firstly, how accurately can automatic rumour verification be performed, and secondly, can we identify features that are characteristic of rumour veracity? In particular, we are interested to know if patterns of support and denial in the conversation around a rumour can be utilised to resolve its veracity.

We address these by developing several machine learning models for rumour verification. We start with non-neural SVM- and RF-based approaches. We follow up by extending the *branch-LSTM* model from chapter 4 to the veracity classification task, as the benefits of using a sequential approach were suggested by previous studies of the rumour stance classification and rumour detection tasks (Zubiaga et al., 2018a, 2017). We then perform a comparison of various feature sets that can aid rumour verification models, analysing their distribution in the dataset and their effect on model performance.

Unlike many previous approaches (Qazvinian et al., 2011; Hamidian and Diab, 2015, 2016), which are using rumours from the same events/topics in both training and testing sets, we focus on making the experimental setup close to the real-world scenario. To accommodate this we use the PHEME dataset in which

rumour stories arise from different major crisis events. We use this basis to split the dataset into subsets corresponding to those events, which are naturally of different sizes and with different class balances. This is in contrast to many existing balanced rumour datasets (Ma et al., 2017; Liu et al., 2015), which obtain already verified rumours from rumour debunking websites and can do so with a similar number of instances in each of the classes. We use this real-world property of our dataset to set up a realistic evaluation scheme: leave-one-event-out (LOEO) cross-validation (CV). This means keeping events in separate folds for cross-validation and always testing the model on events unseen during training, with new topics and vocabulary, making it very challenging.

The chapter is organised as follows. We first describe the datasets (section 5.2) and methods (section 5.3) used, then we define the potentially relevant feature sets and analyse their distribution with respect to the classes (section 5.4). Section 5.5 describes the experimental set up. Finally we present the results of our experiments on the PHEME, RumourEval 2017 and 2019 datasets (section 5.6).

In this chapter we make the following contributions:

- Develop and compare several supervised learning models for rumour verification.
- Study the effects of incorporating different types of features into rumour verification models.
- We provide a baseline rumour verification model for the RumourEval 2019 shared task. We then demonstrate the performance of our LSTM-based model, developed in chapter 4, against other participating systems.

5.2 Data

In this chapter we test different approaches to veracity classification on the PHEME, RumourEval 2017 and RumourEval 2019 datasets. The details about the data collection process, number of instances, class balance and evaluation procedure (training/testing split or cross-validation) are provided in chapter 3. When working with the PHEME dataset we choose to perform experiments in two settings: (1) with the 5 largest events, and (2) with all 9 events, that provide us with two different datasets because of the significant differences between large and small events. The five largest events have a more balanced distribution of *true*, *false* and *unverified* rumours within the event, while the four smallest events usually have a very strong

imbalance towards the majority class, as they often only contain a few rumour stories.

5.3 Methods

In this section we describe various models that we tested for rumour veracity classification.

5.3.1 Non-neural models

The non-neural models described in this section treat veracity classification as a 3-way classification of source tweets introducing the conversation tree. Contextual information about the conversation tree may be added in the form of extra features.

Majority vote is a strong baseline that always predicts the majority class and results in relatively high accuracy due to the class imbalance in the veracity classification task.

Support Vector Machines (SVM) and Random Forest (RF) as described in chapter 2, were used as models for rumour verification using the representation of source tweets as an input. We employed linear kernel SVM in our experiments. When performing experiments with various feature sets we simply concatenated the representation of a source tweet as an average of word vectors with additional features.

5.3.2 NileTMRG

NileTMRG (Enayet and El-Beltagy, 2017) is the winning model of the veracity classification task of the RumourEval 2017 competition. The *NileTMRG* model is based on a Support Vector Machine with Linear Kernel that uses a Bag-of-Words representation of the source tweet conveying a rumour, concatenated with selected features: presence of URL, presence of hashtag and proportion of supporting, denying and querying tweets in the conversation. As information on the proportion of supporting, denying and querying tweets in the conversation is not available for all datasets, the *NileTMRG* model depends on the performance of a stance classification system. We have made our own implementation of the *NileTMRG* model (denoted *NileTMRG**) based on their description (Enayet and El-Beltagy, 2017), swapping the stance classification system proposed by the authors with a better performing model, specifically our *branch-LSTM* model described in the previous

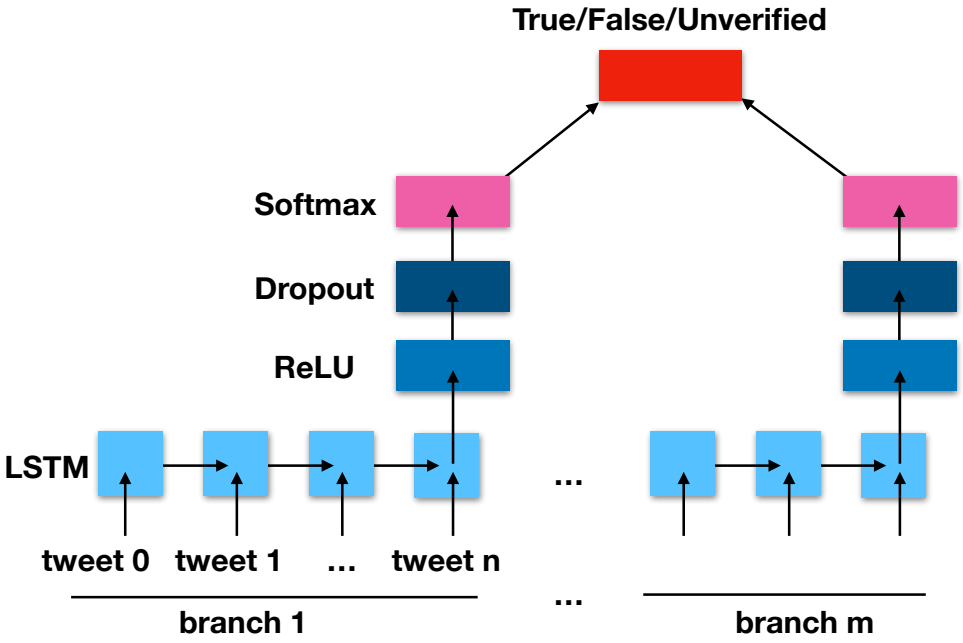


Figure 5.1: Architecture of the *branch-LSTM* model.

chapter 4. The use of neural stance classification component makes NileTMRG* also a neural verification model.

The *NileTMRG* model reflects a pipeline architecture where tasks are performed sequentially and the outcome of the previous step (stance classification) is the input to the next one (veracity classification). The *NileTMRG* model has set a precedent in demonstrating a model that successfully utilises patterns of support and denial for rumour veracity classification.

5.3.3 Branch-LSTM for rumour verification

Given the success of the sequential approach and *branch-LSTM* model on the stance classification task, we continued experiments with the *branch-LSTM* model, extending it to the veracity classification task. The model architecture is illustrated in figure 5.1. It uses the same input structure as for the stance classification *branch-LSTM*, described in section 4.3.3, but it produces a single output per-branch. The veracity prediction for the conversation is then decided using aggregation of per-branch outcomes. While, there are many ways to aggregate per-branch outcomes, from simple majority voting or averaging of softmax predictions, to more complex,

involving the meta-model that does the aggregation, here, we chose to aggregate them by using simple majority voting, and the exploration of the effects of aggregation method is left to the future work.

We describe this model mathematically using the notation defined in sections 4.3.1–4.3.3. An input to the model is a branch z_{mk}, $m \in [1, .., M_k]$, which has a length T_{mk}, and y_k is a veracity label of a conversation that a branch belongs to, one-hot encoded:

$$h_{0t} = \text{LSTM}(z_{mk}); \quad t \in [1, \ldots, T_{mk}] \tag{5.1}$$

$$h_1 = \text{ReLU}(W_1 \cdot h_{0T_{mk}} + b_1); \tag{5.2}$$

$$\ldots$$

$$h_\lambda = \text{ReLU}(W_\lambda \cdot h_{\lambda-1} + b_\lambda); \tag{5.3}$$

$$h_{\lambda+1} = h_\lambda \odot \sigma_{out}; \tag{5.4}$$

$$h_{out} = \text{softmax}(W_{out} \cdot h_{\lambda+1} + b_{out}); \tag{5.5}$$

$$\text{loss} = \sum_{c=1}^{C} y_k^c \log(h_{out}^c), \tag{5.6}$$

where the LSTM is as defined in equations 2.13–2.17 that uses hyperbolic tangent activation function (the default option in the Keras library (Chollet et al., 2015)). Additionally, the recurrent dropout with $p = 0.2$ is applied at the LSTM layer as described in Gal and Ghahramani (2016a). Loss here is defined per-branch, then later it is averaged over all of the branches in the training set (or mini-batch). Predictions for each conversation are then obtained by using majority voting over branch predictions in that conversation.

5.4 Feature analysis and selection

We have studied the effect of different feature types on the performance of rumour veracity classification model. Where there is an overlap with the features defined in section 4.5, the features are defined similarly. The list of features was formed using experience gained from previous experiments with the PHEME dataset on the stance classification task, as well as features used in previous research on rumour detection, verification and other Twitter studies, as described in section 2.2.5. The list of features and their groups are presented in table 5.1. The full description of each feature as well as plots of feature distributions can be found in appendix A.

Feature group	Features
Text	Average of word embeddings
Attachments	URL attached
	Image attached
Lexicon	Use of swear words
	Use of negation
	Use of 'wh-' words
	Use of 'rumour' and 'unconfirmed'
	Use of synonyms of 'false'
	Use of antonyms of 'false'
	Presence of words from existing lexicons
Content formatting	Presence of question mark
	Presence of exclamation mark
	Presence of period
	Presence of hashtag
	Character count
	Word count
	Ratio of capital letters
User features	Number of followers
	Follow ratio
	Account age
	Statuses count
	Verified user
	User has URL
	Geolocation enabled
Social interactions	Favourite count
	Retweet count
Stance	Proportion
	Labels
Tree	Similarity to the source tweet
	Similarity to other tweets in the tree

Table 5.1: List of features, grouped by type, for result tables. Lexicon features were taken from (Hu and Liu, 2004; Kiritchenko et al., 2014; Mohammad, 2012; Mohammad et al., 2009; Nielsen, 2011; Zhu et al., 2014)

We have extracted the described features from the source tweets of the conversations in PHEME (all 9 events) and RumourEval 2017 datasets. Distributions of various binary and non-binary features, across three classes, grouped by feature type are shown in the Appendix A figures A.1 – A.11, for PHEME (shown in figures A.1, A.3, A.5 A.7, A.8, A.10) and RumourEval 2017 (shown in figures A.2, A.4, A.6, A.7, A.9, A.11). Bar plots show percentage of the source tweets or users possessing a feature in each of the classes. Violin plots show per-class density distribution plots for count-based features. We made the following observations:

- Content formatting (figures A.1 and A.2)
  - In the PHEME dataset a question mark is most often present in source tweets from the *unverified* class, while in RumourEval 2017 it is equally frequent in both the *unverified* and *false* classes.
  - Use of hashtags is higher in *unverified* rumours in both datasets.
  - Exclamation mark is not a very frequent feature of source tweets conveying rumour, but it occurs slightly more in *false* tweets, as presumably those rely more on evoking emotion from the reader.
- Attachment (figures A.3 and A.4)
  - *True* rumours have a higher percentage of images and URLs attached than for *false* and *unverified* rumours for both RumourEval 2017and PHEME datasets.
  - The difference between the percentage of *false* and *unverified* rumours varies between RumourEval 2017 and PHEME.
- Social interactions (figures A.5 and A.6)
  - In both datasets *true* rumours have a higher number of retweets.
  - In RumourEval 2017 the *false* class gets a higher number of favourites, while in PHEME the *true* class has the highest numbers of favourites.
- User (figures A.8 and A.9)
  - While a lot of users have a URL attached to the profile, it is more prevalent for users who post *true* rumours.
  - Geolocation seems to have low levels for users for all categories, but it is surprisingly highest for *false* rumours.
  - *True* rumours have the highest percentage of verified users posting them.

- Lexicon (figures A.10 and A.11)
  - The words 'rumour' and 'unconfirmed' are rarely mentioned explicitly in the dataset and are indicators of *false* of *unconfirmed* rumours.
  - Use of negation is slightly increased in *false* and *unverified* rumours comparing to *true* rumours.
- Cosine similarity (figure A.7)
  - There is difference between PHEME and RumourEval 2017 distributions, which is especially notable for the *false* class which has a different interval of values between the datasets.

It is hard to say what effect the difference in density distribution of count-based features will have, therefore it is worth investigating their influence in conjunction with classifier performance.

5.5 Experimental setup

5.5.1 Preprocessing

For the *branch-LSTM* model the preprocessing routine is the same as for he stance classification task, except that stance classification aims to produce a label for each tweet, whereas rumour verification needs to predicts a label for the whole conversation. Text preprocessing consists of the following steps: removal of non-alphabetic characters, converting all words to lower case and tokenising texts. Preprocessing is followed by feature extraction. After features are extracted from each tweet they are concatenated to form an input vector representing a tweet.

Preprocessing for *NileTMRG* differs to the above, as we follow the description in Enayet and El-Beltagy (2017): tokenisation is performed using NLTK TweetTokenizer (Loper and Bird, 2002), and it is followed by a removal of stop words, punctuation characters and twitter-specific words, such as 'rt' and 'via'. This model only uses source tweets (we use CountVectorizer from the sklearn package (Pedregosa et al., 2011) to extract the Bag-of-words representation) and the only information about the conversation is provided in features reflecting the proportion of different stance labels in the conversation.

The *SVM* and *RF* models use the same representation as the *branch-LSTM* model however, similarly to *NileTMRG*, they only use the source tweet as they are not capable of representing a sequence of tweets. Thus we omit the word2vec similarity to source tweet feature in non-sequential models. Stance is also represented

differently in sequential and non-sequential models: in *branch-LSTM* we use predicted stance labels as a feature in each tweet as a label encoded categorical feature; in *NileTMRG*, *SVM* and *RF*, we use the proportion of stances in a conversation.

5.5.2 Evaluation setup

The setup for evaluation is also very similar to the one described for rumour stance classification and is chosen according to the dataset used. For the RumourEval 2017 and RumourEval 2019 datasets we use the split into training, development and testing sets. We perform leave-one-event-out cross-validation on the PHEME dataset. We perform experiments in two settings: (1) with the 5 largest events with the PHEME dataset, and (2) with all 9 events. The five largest events have more balanced distribution of *true*, *false* and *unverified*, within the event, while the four smallest events usually have a very strong majority class representation, as they often only contain a few rumour stories. Thus these set ups provide different levels of difficulty for the model.

To select the hyper-parameters, we use one of the events ('Charlie Hebdo') as the development set, another ('Germanwings crash') as testing and the rest as training. When the hyper-parameters are chosen, we perform leave-one-event-out cross-validation.

5.5.3 Hyper-parameters

In experiments with *NileTMRG*, *SVM* and *RF* models, we used the default parameters in the sklearn package (Pedregosa et al., 2011).

The parameter search algorithm and search space are the same as for the stance classification task. The set of hyper-parameters includes: number of layers, number of nodes in each layer, mini-batch size, number of epochs, learning rate and strength of L2-regularisation. We determine the optimal set of hyper-parameters via testing the performance of the model on the development set for different parameter combinations. We use the Tree of Parzen Estimators (TPE) algorithm to search the parameter space, which is defined as follows: the number of dense ReLU layers varies from one to five; the number of LSTM layers is one or two; the mini-batch size is either 32 or 64; the number of units in the ReLU layer is one of {50, 100, 200, 300, 400, 500, 600}, and in the LSTM layer one of {100, 200, 300, 400}; learning rate is one of $\{10^{-4}, 3 \cdot 10^{-4}, 10^{-3}\}$, the strength of the L2 regularisation is one of $\{0.0, 10^{-4}, 3 \cdot 10^{-4}, 10^{-3}\}$ and the number of epochs is selected from {20, 30, 50,100}. These ranges were chosen manually based on prior experience of the author

	Majority		SVM		RF		branch-LSTM	
	Accuracy	Macro-F	Accuracy	Macro-F	Accuracy	Macro-F	Accuracy	Macro-F
RumourEval 2017	0.286	0.148	0.464	0.473	0.357	0.319	**0.500**	**0.491**
PHEME 5 events	**0.511**	0.226	0.423	**0.336**	0.357	0.269	0.454	**0.336**
PHEME 9 events	**0.444**	0.205	0.339	**0.306**	0.324	0.275	0.314	0.259

Table 5.2: Performance of *SVM*, *RF* and *branch-LSTM* models against majority baseline on RumourEval 2017, and on the PHEME datasets, using 5 the largest events and all 9 events. All of the models use textual features only.

and are similar to typical values in works using deep learning models (Rocktäschel et al., 2015; Goodfellow et al., 2016). We perform 100 trials of different parameter combinations for each of the models.

We choose the best set of hyper-parameters by assessing the minimum loss on the development set. Loss is defined as (1 – macro-averaged F score) in experiments with all of the datasets.

5.5.4 Evaluation metrics

In the results section we present model performance expressed as accuracy, micro- and macro-averaged F1 score, which are calculated as described in section 2.1.3. In the RumourEval 2019 competition, not only class predictions but also confidence scores were considered. If the confidence score was lower than 0.5 then the prediction would get treated as *unverified* regardless of the actual model prediction. Macro-averaged F score was used as the main competition metric. We use the task scoring system to evaluate performance on RumourEval 2019 dataset (Gorrell et al., 2019). For the confidence score, a root mean squared error (RMSE) was calculated as described in section 2.1.3.2.

5.6 Results

We first present the results of the models described above using the representation of each tweet as an average of word2vec word vectors. Then we will present the results showing the effects of incorporating additional feature sets on the performance of the *SVM*, *RF* and *branch-LSTM* models. Finally, we present the results of our model as a baseline for the most recent RumourEval 2019 task and compare it to the works by other participants.

Table 5.2 shows performance, in terms of accuracy and macro-F score, of the *SVM*, *RF* and *branch-LSTM* models against a majority baseline, which always predicts the majority class. We have performed experiments on the RumourEval

	SVM		RF		branch-LSTM	
	Accuracy	Macro F	Accuracy	Macro F	Accuracy	Macro F
Ebola-essien	0	0	0.071	0.044	0.286	0.222
Ferguson	0.213	0.135	0.189	0.117	0.027	0.018
Gurlitt	0.098	0.085	0.213	0.118	0.492	0.439
Ottawashooting	0.509	0.316	0.428	0.310	0.343	0.224
Prince-toronto	0.031	0.041	0.066	0.056	0.205	0.116
Putinmisisng	0.341	0.175	0.182	0.127	0.056	0.050
Sydneysiege	0.416	0.316	0.391	0.279	0.485	0.344
Charliehebdo	0.329	0.300	0.416	0.330	0.342	0.247
Germanwings-crash	0.387	0.381	0.332	0.305	0.369	0.303
Total	0.339	0.306	0.324	0.275	0.314	0.259

Table 5.3: Per-event results of *SVM*, *RF* and *branch-LSTM* models using textual features on the full PHEME dataset with 9 events.

	SVM	RF	branch-LSTM
True	0.454	0.465	0.465
False	0.149	0.136	0.240
Unverified	0.317	0.225	0.072

Table 5.4: Per-class performance in terms of macro-F score of *SVM*, *RF* and *branch-LSTM* models using textual features on the full PHEME dataset with 9 events.

2017 dataset using its training/development/testing split and use leave-one-event-out cross-validation on the PHEME dataset in two settings: using the 5 largest events and using all 9 events. The importance of having two separate settings for the PHEME dataset is that five largest events represent very different situations to the four smallest ones. The five largest events are major crisis situations that are discussed widely and provoke a lot of rumours from all three classes, while the four smallest events are a lot more focused on one of two rumour stories with a very strong majority class representation (see table 3.2 in chapter 3). As a result, models in a leave-one-event cross-validation set up tend to either achieve very high performance on the small events or very low. This can be seen in table 5.3. Interestingly, while 'Ferguson' is not a one of the small events, performance on it is rather low. We believe this is due to the difference in class balance between 'Ferguson', which is strongly dominated by the *unverified* class, and the events that the model was trained on. Table 5.4 shows per-class performance for each of the models. As *true* is the majority class, all of the models perform well on it, and while *false* is the hardest to predict class for *SVM* and *RF* models, *branch-LSTM* struggles with *unverified* rumours.

The difference between accuracy and macro-F score for the majority base-

line is due to a class imbalance in all of the three datasets we are using. All of the proposed models outperform the majority baseline in terms of macro-F score, however the majority baseline has the highest accuracy in the PHEME 5 events and PHEME 9 events experiments. As we are using leave-one-event-out cross-validation on the PHEME dataset, this means that the models have not been exposed to the vocabulary of the events they are being tested on, which is a realistic and very challenging scenario. While in the RumourEval 2017 dataset, the testing set contains new rumour stories about the same events that were used for training and development, along with two completely unseen events, which makes it a somewhat easier scenario for models to tackle. On the RumourEval 2017 dataset,*branch-LSTM* outperforms both *SVM* and *RF* models. However, the SVM-based *NileTMRG* model (Enayet and El-Beltagy, 2017) (described in section 5.3.1) achieves an even higher accuracy score of 0.536 on the RumourEval 2017 testing set. We attribute the success of this model to the features used, in particular the proportion of stance labels in the conversation tree. Thus our implementation *NileTMRG** that uses the *branch-LSTM* model for stance label prediction, which is superior to the one originally used, improves the accuracy up to 0.570 with macro-F score of 0.539.

On the PHEME dataset with 5 events, *branch-LSTM* also performs better than other models. However, on the full PHEME dataset with 9 events, *branch-LSTM* is outperformed by both *SVM* and *RF*, with *SVM* achieving the highest scores. In general, accuracy and macro-F scores for all of the models drop when using the full dataset with 9 events as opposed to only 5 events, even though the models are getting more training data. This highlights how difficult the task of rumour verification is, in particular when imitating a realistic scenario with leave-one-event-out cross-validation. To highlight this point even further, we performed experiments with a simple SVM model, where we split the conversations into cross-validation folds randomly, without regard to the event they belong to. We split PHEME 5 events into 5 folds and PHEME 9 events into 9 folds to preserve the number of folds in experiments. As a result of these experiments, the accuracy of the *SVM* model, using textual features only, rises from 0.423 to 0.739 on PHEME with 5 events, and from 0.339 to 0.766 on the full PHEME with 9 events. While we did not explicitly test it on *RF* or *branch-LSTM* models, we would expect to see similar pattern of performance increase when the model is presented with event data and thus its vocabulary at training time.

Feature set	SVM		Random Forest		branch-LSTM	
	Accuracy	Macro-F	Accuracy	Macro-F	Accuracy	Macro-F
text	0.464	0.473	0.357	0.319	0.500	0.491
text+attachments	0.464	0.457	**0.571**	**0.554**	0.481	0.466
text+interactions	0.357	0.175	0.464	0.451	0.519	0.504
text+lexicon	0.429	0.413	0.285	0.247	0.519	0.506
text+punctuation	0.536	0.506	0.429	0.420	0.444	0.421
text+stance	0.464	0.473	0.464	0.441	0.444	0.417
text+user	0.393	0.193	0.500	0.474	0.519	0.488
text+tree	0.464	0.473	0.429	0.409	0.444	0.410
all features	0.392	0.347	0.536	0.527	0.481	0.470

Table 5.5: Effects of incorporating additional features on the performance of *SVM*, *RF* and *branch-LSTM* models on the RumourEval 2017 dataset.

Feature set	SVM		Random Forest		branch-LSTM	
	Accuracy	Macro-F	Accuracy	Macro-F	Accuracy	Macro-F
text	0.423	0.336	0.357	0.269	0.454	0.336
text+attachments	0.417	0.329	0.373	0.279	0.452	0.324
text+interactions	0.411	0.389	0.376	0.279	**0.462**	**0.340**
text+lexicon	**0.449**	**0.354**	0.396	0.298	0.460	0.329
text+punctuation	0.416	0.297	0.363	0.267	0.448	0.327
text+stance	0.419	0.332	0.377	0.284	0.447	0.330
text+user	0.276	0.255	0.367	0.274	0.441	0.317
text+tree	0.424	0.338	0.367	0.269	0.444	0.325
all features	0.263	0.234	0.381	0.274	0.451	0.332

Table 5.6: Effects of incorporating additional features on the performance of *SVM*, *RF* and *branch-LSTM* models on the five largest events in the PHEME dataset.

Feature set	SVM		Random Forest		branch-LSTM	
	Accuracy	Macro-F	Accuracy	Macro-F	Accuracy	Macro-F
text	0.339	0.306	0.324	0.275	0.313	0.259
text+attachments	0.334	0.301	0.321	0.267	0.257	0.235
text+interactions	0.252	0.225	0.328	0.276	0.250	0.223
text+lexicon	**0.352**	**0.310**	0.328	0.276	0.298	0.287
text+punctuation	0.330	0.269	0.311	0.269	0.185	0.182
text+stance	0.337	0.304	0.320	0.273	**0.352**	**0.351**
text+user	0.318	0.287	0.304	0.261	0.173	0.157
text+tree	0.341	0.308	0.331	0.288	0.262	0.222
all features	0.231	0.229	0.300	0.245	0.247	0.17

Table 5.7: Effects of incorporating additional features on the performance of *SVM*, *RF* and *branch-LSTM* models on the full PHEME dataset.

5.6.1 Relevant feature selection experiments

Tables 5.5, 5.6 and 5.7 present the effects of incorporating additional features on the performance of the *SVM*, *RF* and *branch-LSTM* models on RumourEval 2017, PHEME 5 events and PHEME 9 events datasets respectively. Overall, there is no clear 'winner' among the feature sets that would lead to an improved performance across all of the models and all of the datasets. Also, different classifiers make better use of different feature sets. The addition of extra features does not necessarily lead to improvement over purely textual features.

In the experiments with the RumourEval 2017 dataset, the best score is reached by the *RF* classifier using text+attachments features, which outperforms the accuracy and macro-F scores achieved by *NileTMRG**. The *SVM* model makes use of punctuation features, reaching the same accuracy score as the *NileTMRG* model, while *branch-LSTM* reaches its highest accuracy scores with interactions, lexicon and user features.

In the experiments with the PHEME dataset, the *RF* classifier consistently performs worse than *SVM* and *branch-LSTM*. In the experiments using the 5 largest events, *branch-LSTM* reaches the highest accuracy of 0.462 using text+interactions features, while *SVM* acheives the highest macro-F score of 0.354 using text+lexicon features. Lexicon-based features also lead to the best performing *SVM* model on all 9 events from the PHEME dataset. However, the *branch-LSTM* model that uses stance as features has matching performance in terms of accuracy but with a better macro-F score. If we calculate average of accuracy and macro F-scores across all of the datasets, the best performing model in terms of macro F-score is *branch-LSTM* with text+lexicon features (accuracy - 0.426, macro F-score - 0.374), and *SVM* with text+punctuation features in terms of accuracy (accuracy - 0.427, macro F-score - 0.357).

While not universally true, the neural sequential approach with the *branch-LSTM* model outperforms the *RF* and *SVM* models in the majority of our experiments (13 wins out of 18 experiments comparing with *SVM* and *RF*). Also, if we calculate average of accuracy and macro F-scores across all of the datasets and feature sets, *branch-LSTM* has the highest performance (accuracy - 0.398, macro F-score - 0.341), followed by *SVM* (accuracy - 0.381, macro F-score - 0.330), and *RF* taking the third place (accuracy - 0.380, macro F-score - 0.325). This points us to the potential benefits of using a sequential approach for the rumour verification task.

We find that it is hard to draw definite conclusions about which features are indicative of rumour veracity from our experiments measuring the effects of us-

Rank	System	Macro-F score	RMSE
1	eventAI	0.5765	0.6078
*	branchLSTM	0.3364	0.7806
* (late submission)	FINKI NLP	0.3326	0.6846
*	NileTMRG	0.3089	0.7698
2	WeST (CLEARumor)	0.2856	0.7642
*	Majority	0.2241	0.7115

Table 5.8: Veracity classification leaderboard for the RumourEval 2019 competition.

	Accuracy	Macro-F	RMSE	True	False	Unverified
branch-LSTM	0.383	0.336	0.781	0.314	0.529	0.167
NileTMRG*	0.407	0.309	0.769	0.245	0.557	0.125

Table 5.9: Performance of *branch-LSTM* and *NileTMRG** baselines on the RumourEval 2019 dataset.

ing various feature sets. If we calculate average of accuracy and macro F-scores across all of the datasets and models, we find that text+attachments feature set leads to highest performance (accuracy - 0.408, macro F-score - 0.357), followed by text+stance (accuracy - 0.403, macro F-score - 0.356), and text only features on the third place (accuracy - 0.392, macro F-score - 0.340). The experiments show how widely rumours from different events and datasets differ from each other and if performed with each dataset in isolation would lead to completely different conclusions about the nature of the rumours. This highlights the complexity of the rumour classification problem and the need for further model and dataset development in order to combat it.

Moore and Rayson (2018) found that a lot of studies reporting advances in the field of target dependent sentiment analysis lack comparability and generalisability in train, test or validation data, and hence recommended that future experiments should consider a variety of datasets, as well as documenting and releasing their methods to improve both repeatability and generalisability. In line with the conclusions of Moore and Rayson (2018), we also consider that it is crucial, particularly in rumour verification field, to use a realistic setup of testing on unseen rumours and verifying findings on multiple datasets.

5.6.2 RumourEval 2019 shared task

As the *branch-LSTM* model for stance classification was used as a baseline for the RumourEval 2019 task, we have extended it to the veracity classification task as well. As we did not define a unique feature set leading to best performance

across all models and datasets, we have decided to use the same feature set for both tasks (the feature set is listed in the section 4.7.4). Table 5.8 shows a partial leaderboard of the competition. Macro-averaged F score was the competition metric. Baseline systems were outperformed by the winning system eventAI (outperforms both baselines) and a late submission by FINKI NLP (outperforms *NileTMRG*, but a reaches similar result to *branch-LSTM*). Table 5.9 shows the performance of both baselines in terms of accuracy, RMSE and macro-averaged F score, and also per-class F score. While *branch-LSTM* achieves a higher F score and gets a higher place in the leaderboard, *NileTMRG** achieves a higher accuracy score. Both systems perform well on identifying *false* rumours.

The top system on the leaderboard, eventAI (Li et al., 2019b), is an ensemble of classifiers: SVM, Random Forest, Logistic Regression and a Neural Network with three connected layers, where individual post representations are created using an LSTM with attention (Rocktäschel et al., 2015; Xu et al., 2015). Whilst not being sequential in nature, this system shows drastically higher performance than the *branch-LSTM* baseline and systems submitted by other participants. Ensemble systems have been shown to outperform individual systems in many cases (Woźniak et al., 2014; Britto Jr et al., 2014) due to their ability to improve predictions by decreasing variance, bias and preventing overfitting. However, as training of multiple independent models is required, the use of ensemble learning is often computationally expensive, depending on what kind of models are included in the ensemble.

5.7 Conclusions

In this chapter we have studied automated rumour verification using Twitter conversation trees as a 3-way classification task. We have developed and compared several approaches to the task: non-neural, which mainly focuses on the source tweet conveying the rumour, and a neural sequential approach that models branches composed of the source tweet and the responses following it.

Results of testing the models corresponding to these approaches on three qualitatively different datasets have shown that a sequential neural approach performs better than *SVM* and *RF* models in most of our tests. This is not universally the case, as sometimes *SVM* is able to outperform *branch-LSTM*, however usually *branch-LSTM* is preferred for performance, which suggests the importance of utilising context provided by the conversation.

We have performed extensive experiments investigating the effects of incorporating additional features into rumour verification models. We were able to suc-

cessfully identify features which benefit specific models on each of the datasets. However, as many of the feature sets benefited some of the models on one or more of the datasets, but not consistently across models and datasets, this does not lead us to a definitive answer on which features are the most indicative of rumour veracity overall. In fact, this suggests that such a universally optimal feature set may not exist, as we find that feature sets explored so far more attuned to the details of the datasets and events therein perform better than a 'one size fits all' feature set.

We were particularly interested in the impact of utilising patterns of support and denial across responses on model performance. We have shown that the stance of responses, expressed as either labels in sequential models or a proportion in non-sequential ones, improves classification performance of both sequential (*branch-LSTM* on the full PHEME dataset) and non-sequential (*NileTMRG* (Enayet and El-Beltagy, 2017) and *NileTMRG** on RumourEval 2017) models. Therefore we can conclude that stance has the potential of being an indicative feature. This leads us to consider the need for finding other ways of incorporating further information into rumour verification models. This is explored in the subsequent chapter 6.

In summary, this chapter highlights the complexity of the problem of rumour spread as it can cover various topics, very different time spans, sources and audiences.

Each rumour introduces a new domain that presents a very difficult challenge. In order to address which, large, professionally annotated datasets with wide coverage of topics can be created, as well as new cross-domain models should be developed, for example, by identifying domain-independent features indicating the veracity of a rumour. Then, the selected set of indicative features should be tested across datasets and domains to confirm that they are not representing spurious correlations in the small data. In general, the problem of rumour spread requires further understanding of rumours and the properties of their spread, as well as the specific nuances of the platforms where they are circulating. We also consider that it is very important for future studies in the rumour verification field to use a realistic setup of testing on unseen rumours and verifying findings on multiple datasets.

CHAPTER 6

Multitask Learning for Rumour Verification

6.1 Introduction

As discussed in chapter 2.2.1 the rumour resolution process can be defined as a pipeline involving four sub-tasks (Zubiaga et al., 2018c): rumour detection, rumour tracking, stance classification, and rumour verification. In previous research studies each of these four tasks have been addressed separately (Lukasik et al., 2016; Liu et al., 2016; Enayet and El-Beltagy, 2017). However, the way these subtasks interact and their integration into a complete rumour resolution system is yet to be explored. In this chapter we address research question RQ7 as outlined in chapter 1, exploring the ways to leverage the interaction between the tasks in a rumour resolution pipeline. We express the rumour resolution process as a multi-task problem that needs to address a number of challenges. We consider veracity to be the main task and the rest of the components are auxiliary tasks that can be leveraged to boost the performance of the veracity classifier.

We propose to achieve this by using a multi-task learning approach. Multi-task learning (Caruana, 1998) refers to the joint training of multiple tasks, which has gained popularity in recent years and has helped to improve performance in a number of different tasks and machine learning architectures in Machine Learning and Natural Language Processing (Collobert and Weston, 2008). Its effectiveness is mainly attributed to learning shared representations of closely related tasks, such that two complementary tasks can give each other 'hints'.

We assess the effectiveness of using a multi-task learning approach for the rumour resolution process by comparing four different scenarios: (1) performing single task learning for veracity classification, (2) performing multi-task learning that combines stance and veracity classification with the aim of boosting performance of

the latter, (3) performing multi-task learning that combines rumour detection and verification, comparing the improvement brought by each of the two auxiliary tasks: rumour detection versus rumour stance classification, and (4) performing multi-task learning that combines rumour detection, stance classification and veracity prediction to improve the performance for veracity.

Our results show that a multi-task learning scenario that leverages all three subtasks, where veracity classification is the main task, while stance classification and rumour detection are auxiliary tasks, leads to substantial improvements over a standard, single task veracity classification systems, as well as a majority baseline. The combination of all three subtasks also outperforms the multi-task learning scenario where only two of the subtasks are combined.

We also compare two architectures for multi-task learning: simple shared layer based architecture and sluice network (Ruder et al., 2017), where trainable parameters control the amount of sharing. Our results show that while both architectures lead to improvements over single-learning approach, in most cases we tested the simple shared layer based architecture is more beneficial.

Additionally, we test the generalisability of the approach on the Emergent dataset that is similar in topic, as it is concerned with fake claims made in news articles, and is also annotated for the tasks of stance and veracity classification. However, it contains news article headlines rather than social media posts.

Further, we perform experiments with the addition of extra features (feature categories described in section 5.4) to the multi-task learning model in order to evaluate their effect in this setting.

To summarise, in this chapter we make the following contributions:

- We propose to leverage the relation between the tasks in a rumour resolution pipeline using multi-task learning.
- We develop several deep learning models that implement the multi-task learning approach and test them on several datasets.
- We show that multi-task learning is beneficial compared to single-task learning with a similar model for the main task of veracity classification. We also find that a combination of three tasks from the verification pipeline shows superior results to the combination of pairs of tasks.
- We perform experiments evaluating the effect of incorporating additional features into the multi-task learning model and find that textual features are sufficient to reach its peak performance.

6.2 Related work

Multi-task learning refers to the joint learning of several related tasks with a shared representation. In recent years, a multi-task learning approach has been successfully applied in combination with neural networks for a variety of NLP tasks (Collobert and Weston, 2008).

There are two major approaches to parameter sharing: hard and soft. Hard parameter sharing implies that different tasks are using the same hidden layer(s). Hard parameter sharing greatly reduces the risk of overfitting. Soft sharing means that each task has its own model, but the distance between the parameters of the models is regularised with respect to the chosen norm in order to encourage the parameters to be similar (Yang and Hospedales, 2016; Abu-Mostafa, 1990).

In this work we use the most common approach to multi-task learning, namely hard parameter sharing.

The effectiveness of a multi-task learning approach is attributed to: effectively increasing the size of the training set by using additional datasets for related tasks and regularisation, as the model has to learn a shared representation for multiple tasks there is less risk of overfitting on one of them.

In multi-task learning auxiliary tasks can be used to direct the main task to use/learn the features that it otherwise would have ignored or failed to identify due to complex relations between the tasks and features. For example, multi-task learning is particularly useful when potentially helpful features are not directly used as such for the main task, but are instead 'suggested' by the auxiliary task, e.g. predicted as labels in the auxiliary task. This use case is very relevant to this work; stance classification could be used as a feature in a system for veracity classification, as has indeed been the case in previous studies, which have shown a relationship between the two tasks. Zhao et al. (2015) and Enayet and El-Beltagy (2017) have created successful models using this premise. However these studies assume access to stance and veracity labels for the same data, which does not apply in our case as we do not have true stance labels for all of the conversations in the dataset and thus would have to rely on a stance classification system.

While a lot of work reports positive outcomes of the application of multi-task learning to various NLP tasks (Collobert and Weston, 2008; Aguilar et al., 2017; Lan et al., 2017), there are also studies showing that this is not always the case (Alonso and Plank, 2017; Bingel and Søgaard, 2017). Alonso and Plank (2017) were the first to demonstrate that multi-task learning brings benefits only for some combinations of main and auxiliary tasks. They also investigate the relationship

between the multi-task learning outcome and the properties of the dataset. We perform a similar analysis to examine the link between the properties of our rumour datasets and the results of the multi-task learning approach used.

6.3 Data

We use RumourEval 2017 and PHEME datasets (described in section 3) for experiments in this chapter. We perform two types of experiments using different subsets of the PHEME dataset: (1) using the five largest events (see table 3.2) and (2) using all nine events. The five largest events create a more balanced dataset as those are major crisis events during which true updates and false information on different aspects of the event were shared and discussed, whereas the 4 smallest events only contain a single rumour story at the core of the event. We also test our model on the Emergent dataset of claims and relevant news articles for rumour veracity and stance classification tasks.

6.4 Models

6.4.1 Multi-task learning approach

We leverage the relationship between the tasks from the rumour classification pipeline in a joint multi-task learning setup. Figure 6.1 illustrates our approach. At the base of it is a sequential approach represented by a shared LSTM layer, which is followed by a number of task-specific layers. The possible task combinations are shown as dotted lines on figure 6.1 that can be present or absent depending on the combination. As we are mainly interested in improved performance on rumour verification task, we perform experiments in three setups: joint training of (1) stance or (2) rumour detection together with veracity classification, and (3) learning all three tasks together. The cost function in the multi-task models is a sum of losses from each of the tasks. Datasets for each of the three tasks are not equal in size, therefore when the training instance is lacking a label for one of the tasks, its prediction does not add anything to the loss function, as if it had been predicted correctly.

We describe this model mathematically using the notation defined in sections 4.3.1–4.3.3 and 5.3.3. An input to the model is a branch z_{mk} where $m \in [1, \ldots, M_k]$, which has a length T_{mk}, and y_k^D is the binary label indicating whether a conversation is a rumour, y_k^V is the veracity label of a conversation that a branch belongs to, y_t^S is a stance label of a tweet, and all labels are one-hot encoded. The following

describes a forward propagation pass over our multitask learning model including all three tasks:

$$h_{0t} = \text{LSTM}(z_{mk}); \quad t \in [1, \ldots, T_{mk}] \tag{6.1}$$

$$h^S_{1t} = \text{ReLU}(W^S_1 \cdot h_{0t} + b^S_1); \tag{6.2}$$

$$\ldots$$

$$h^S_{\lambda t} = \text{ReLU}(W^S_{\lambda t} \cdot h^S_{(\lambda-1)t} + b^S_{\lambda t}); \tag{6.3}$$

$$h^S_{(\lambda+1)t} = h^S_{\lambda t} \odot \sigma^S_{out,t}; \tag{6.4}$$

$$h^S_{out,t} = \text{softmax}(W^S_{out,t} \cdot h^S_{(\lambda+1),t} + b^S_{out,t}); \tag{6.5}$$

$$\text{loss}^S = \sum_{t=1}^{T_{mk}} \sum_{c=1}^{C} y^{Sc}_t \log(h^{Sc}_{out,t}) \delta_t \eta^S_t; \tag{6.6}$$

$$h^V_1 = \text{ReLU}(W^V_1 \cdot h_{0T_{mk}} + b^V_1); \tag{6.7}$$

$$\ldots$$

$$h^V_\lambda = \text{ReLU}(W^V_\lambda \cdot h^V_{\lambda-1} + b^V_\lambda); \tag{6.8}$$

$$h^V_{\lambda+1} = h^V_\lambda \odot \sigma^V_{out}; \tag{6.9}$$

$$h^V_{out} = \text{softmax}(W^V_{out} \cdot h^V_{\lambda+1} + b^V_{out}); \tag{6.10}$$

$$\text{loss}^V = \sum_{c=1}^{C} y^{Vc}_k \log(h^{Vc}_{out}) \eta^V_k; \tag{6.11}$$

$$h^D_1 = \text{ReLU}(W^D_1 \cdot h_{0T_{mk}} + b^D_1); \tag{6.12}$$

$$\ldots$$

$$h^D_\lambda = \text{ReLU}(W^D_\lambda \cdot h^D_{\lambda-1} + b^D_\lambda); \tag{6.13}$$

$$h^D_{\lambda+1} = h_\lambda \odot \sigma_{out}; \tag{6.14}$$

$$h^D_{out} = \text{softmax}(W^D_{out} \cdot h^D_{\lambda+1} + b^D_{out}); \tag{6.15}$$

$$\text{loss}^D = \sum_{c=1}^{C} y^{Dc}_k \log(h^{Dc}_{out}) \eta^D_k; \tag{6.16}$$

$$\text{loss} = \alpha^S \text{loss}^S + \alpha^V \text{loss}^V + \alpha^D \text{loss}^D, \tag{6.17}$$

where S stands for stance, V for veracity, D for detection tasks. LSTM layer is as described in 5.3.3 and, while equations show a single shared LSTM layer, the number

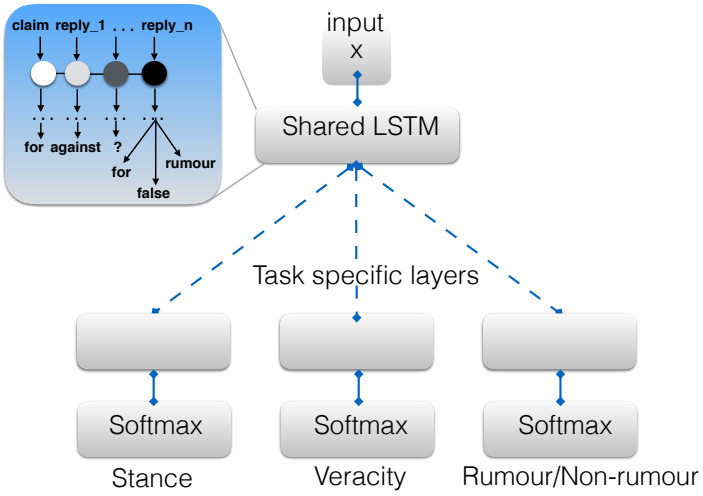


Figure 6.1: Multi-task learning models. Dotted lines represent that same setup is used for learning one, two or three tasks. Input format is a branch of tweets. Stance classification implies predictions per-tweet, whereas detection and verification tasks only require per-branch output.

of LSTM layers is a hyper-parameter that can be varied between 1 and 3. Variables η_t^S, η_k^V, η_k^D are binary flags indicating whether an instance has a true label for the task, such that those instances that are not labelled for either stance or veracity do not contribute to the loss function. Variables α^S, α^V, α^D are the weights that each task can contribute to overall loss, we have performed experiments with equal weights for each task.

6.4.2 Sluice

Additionally to the architecture based on *branch-LSTM* with hard parameter sharing at the LSTM layer, we experiment with a sluice network architecture (Ruder et al., 2017) that controls the amount of sharing between the tasks using a learned parameter. Mathematical notation for this model can be found in Ruder et al. (2017). We perform experiments with two tasks: stance and veracity classification. The architecture is shown in figure 6.2. The input to the model is a branch of tweets, which is passed to LSTM layers corresponding to each of the tasks. Each layer is also divided into subspaces, which allow the network to learn task-specific and shared representations. The output of the LSTM layers is then put through a cross-stitch that produces a weighted combination using parameter α. The LSTM layers and cross-stitch procedure are then repeated. After that layer-stitch nodes produce a

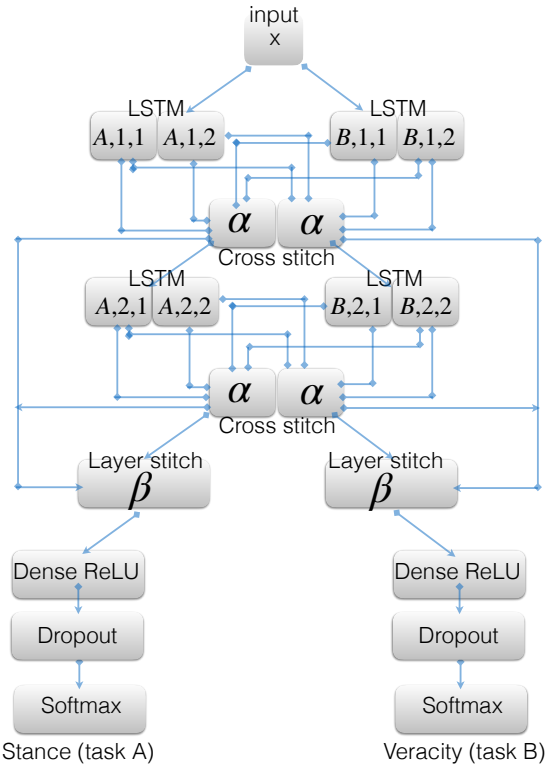


Figure 6.2: Sluice network architecture for stance and veracity classification tasks

weighted combination of both cross-stitch layers using parameter β. The resulting representation goes into task-specific ReLU dense layers, followed by a dropout and a softmax layers to output class probabilities. The loss function is categorical cross entropy for both tasks and the weight of each task is set to 0.5. Hard parameter sharing can be called a case of a sluice network when all α values are set to the same constant. We use the same tweet representation as average of word2vec word vectors as we used in other models.

6.4.3 Baselines

We compare proposed multi-task learning approach with several baselines, models for rumour verification from chapter 5. First of all, majority vote, a strong baseline which results in high accuracy due to the class imbalance in the veracity classification task. Another baseline is NileTMRG (as in section 5.3.1) (Enayet and El-Beltagy,

2017) that is showing the scenario where pipeline tasks are performed sequentially and the outcome of the previous step (stance classification) is an input to the next one (veracity classification). The NileTMRG model sets a precedent in demonstrating that patterns of support are useful indicators of rumour veracity. Finally, we also compare against single task *branch-LSTM* model with textual features only (as in section 5.3.3).

6.4.4 Features

We perform experiments comparing the single-task learning baselines with multitask learning systems using a simple tweet representation: average of word2vec word vectors pre-trained on GoogleNews dataset.

As we establish the benefits of the multitask learning systems, we perform additional experiments with extra features concatenated with the tweet representation to see if they provide further performance improvements. We use the same feature groups as discussed in section 5.4.

6.5 Experiment setup

6.5.1 Hyperparameters

We determined the optimal set of hyper-parameters by testing the performance of our models on the development set for different parameter combinations. We used the Tree of Parzen Estimators (TPE) algorithm[1] to search the parameter space and minimise the loss function expressed as $(1 - macroF)$ for single task, and $(1 - macroF_a)(1 - macroF_b)$ or $(1 - macroF_a)(1 - macroF_b)(1 - macroF_c)$ for multi-task learning of two and three tasks respectively. This loss function gives equal weight to all tasks. The parameter space is defined as follows: the number of dense ReLU layers varies from one to four; the number of LSTM layers is {1, 2}; the mini-batch size is 32; the number of units in the ReLU layer is {300, 400, 500, 600}, and in the LSTM layer is {100, 200, 300}; the strength of the L2 regularisation is $\{10^{-4}, 10^{-3}\}$ and the number of epochs is 50. We performed 30 trials of different parameter combinations optimising for accuracy on the development set in order to choose the best combination. We also use 50% dropout before the output layer. Zero-padding and masks that account for the varying lengths of the input branches were used in all our models. Models were implemented[2] using Python 3 and the Keras package.

[1] We used the implementation of the TPE algorithm in the hyperopt package
[2] https://github.com/kochkinaelena/Multitask4Veracity

6.5.2 Evaluation

The RumourEval dataset was provided with a training/development/testing split. We tune parameters on the development set and then retrain the model on the combined training and development sets before evaluation on the testing set. On the PHEME dataset we perform leave-one-event-out (LOEO) cross-validation, which makes this task setup harder than for RumourEval but closer to the realistic scenario where we want to verify unseen rumours. When choosing parameters, the Charlie Hebdo event was used as the development set as it has balanced labels. We evaluate models using accuracy and macro-averaged F score as the tasks in the PHEME dataset suffer from a class imbalance.

6.6 Results and discussion

The main results of our experiments are presented in table 6.1. It shows the results of the multi-task learning models with two tasks: MTL2 Veracity+Stance and MTL2 Veracity+Detection; MTL3 with three tasks Stance+Veracity+Detection, Majority, NileTMRG* and single task branchLSTM baselines on the main task, veracity classification (single task baselines are taken from previous veracity classification experiments, see table 5.2).

As the datasets contain a significant class imbalance, the majority baseline achieves fairly high accuracy scores. However due to the nature of this task, it is more important for a model to recognize all of the classes, especially *false* rumours, therefore the macro-averaged F score is more important for performance evaluation. All models demonstrate improvement over the majority baseline in terms of macro-F score. We observe improvements of multi-task approaches over single task learning in both accuracy and macro-F score, and adding the third task brings further improvement.

MTL2 Veracity+Detection and MTL3 experiments were performed only on the PHEME dataset as the RumourEval dataset consists only of rumours (no rumour detection). On the RumourEval dataset the single task model branchLSTM outperforms the majority baseline, although it does not perform as well as NileTMRG*, while MTL2 shows improvement over both NileTMRG* and branchLSTM. Experiments on the PHEME dataset also show a pattern of increasing scores: MTL2 outperforms single task models and MTL3 outperforms MTL2. Comparing the performance of the MTL2 model using stance as an auxiliary task with the model using rumour detection as an auxiliary task on the PHEME 5 events dataset we observe that both models bring an improvement over the single task branchLSTM baseline.

RumourEval						
	Majority	**NileTMRG***	**branchLSTM**	**MTL2 Veracity+Stance**	**MTL2 Veracity+Detection**	**MTL3**
Macro F	0.148	0.539	0.491	**0.558**	-	-
Accuracy	0.286	0.570	0.500	**0.571**	-	-
PHEME 5 events						
	Majority	**NileTMRG***	**branchLSTM**	**MTL2 Veracity+Stance**	**MTL2 Veracity+Detection**	**MTL3**
Macro F	0.226	0.339	0.336	0.376	0.373	**0.396**
Accuracy	**0.511**	0.438	0.454	0.441	0.410	0.492
PHEME 9 events						
	Majority	**NileTMRG***	**branchLSTM**	**MTL2 Veracity+Stance**	**MTL2 Veracity+Detection**	**MTL3**
Macro F	0.205	0.297	0.259	0.318	0.345	**0.405**
Accuracy	**0.444**	0.360	0.314	0.357	0.397	0.405

Table 6.1: Comparison of performance of sequential single task approach, multi-task learning approaches with two and three tasks with majority and NileTMRG baselines on veracity classification task. Majority class is *true*.

	MTL2		Sluice	
	Accuracy	Macro F	Accuracy	Macro F
RumourEval 2017	**0.571**	**0.558**	0.444	0.465
PHEME 5 folds	**0.441**	**0.376**	0.402	0.355
PHEME 9 folds	0.357	0.318	**0.362**	**0.331**

Table 6.2: Comparison between plain hard parameter sharing multitask learning setup and sluice model used for joined learning of stance and veracity classification tasks

When using all 9 events of the PHEME dataset we observe worse performance than on the 5 events. Even though we are adding more training data, the 4 additional events are qualitatively different to the 5 large news-breaking events. Each of these 5 large events contained rumours labeled with all classes as well as non-rumours, whereas the 4 additional events are small and the event itself is a *false* or *unverified* rumour. This highlights the difficulty of the rumour verification task in a leave-one-event-out setup and the importance of high quality data. NileTMRG* is a very strong baseline, and while the single task branchLSTM model is competitive when we are using 5 largest events, NileTMRG* is only outperformed by the multi-task learning approach.

6.6.1 Performance of sluice model

Table 6.2 shows comparison of the results in terms of accuracy and macro-averaged F score of sluice architecture and MTL2 architecture with shared LSTM layer between stance and veracity classification tasks on RumourEval 2017, PHEME 5 folds and 9 folds. While sluice has more complex architecture, it outperforms MTL2 only on

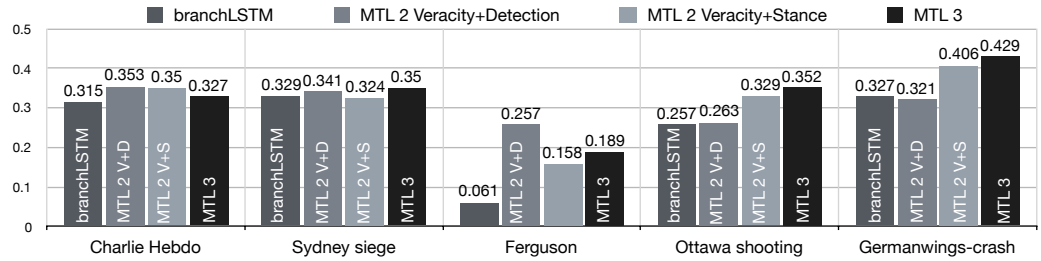


Figure 6.3: Comparison of macro F score of sequential single task approach (branchLSTM), multi-task learning approaches with two and three tasks on different events from the PHEME dataset.

MTL3 5 events	**Macro F**	**Accuracy**	**True**	**False**	**Unverified**
Charlie Hebdo	0.327	0.369	0.502	0.227	0.251
Sydney siege	0.350	0.575	0.731	0.153	0.168
Ferguson	0.189	0.338	0.058	0	0.508
Ottawa shooting	0.352	0.645	0.789	0.168	0.100
Germanwings-crash	0.429	0.420	0.538	0.358	0.364

Table 6.3: Per event and per-class results for multi-task learning approach with 3 tasks on PHEME 5 events.

full PHEME dataset with 9 events in both accuracy and macro F score. Thus we do not continue further experiments with sluice model.

6.6.2 Per-event and per-class results analysis

Here we analyse the performance of proposed models on each of the 5 largest events. Figure 6.3 illustrates the comparison of macro-averaged F scores of the proposed models for each of the events. Multi-task learning models outperform single task learning approaches for each event. The Ferguson event is the hardest one for all of the models (see table 6.3) as it has a different class distribution to all other events (class distribution is shown in table 3.2).

Table 6.3 shows per-event and per-class performance of the multi-task learning model that incorporates all three tasks (MTL3). We have analysed similar performance breakdown tables for other models. All models tend to predict the majority class (*true*) the best. As the Ferguson event is strongly dominated by *unverified* rumours, it is the only event with high performance on the *unverified* class. Single task models (NileTMRG*, branchLSTM) are better at identifying *false* than *unverified* rumours, whereas multitask models (MTL2, MTL3) are better at identifying *unverified* than *false* rumours.

	Kurtosis			Entropy			TTR		
Events	**S**	**V**	**D**	**S**	**V**	**D**	**S**	**V**	**D**
charliehebdo	-0.73	-1.25	-0.18	0.89	1.08	0.53	0.2	0.11	0.07
ottawashooting	-0.83	0.33	-1.99	1.04	0.82	0.69	0.19	0.11	0.09
germanwings-crash	-1.11	-0.86	-1.99	0.99	0.99	0.69	0.26	0.16	0.13
sydneysiege	-0.79	0.71	-1.91	1.01	0.76	0.68	0.19	0.11	0.07
ferguson	-0.5	17.44	-0.64	0.99	0.28	0.56	0.17	0.09	0.06
ebola-essien	-0.22	-3	-3	1.01	0	0	0.38	0.27	0.27
putinmissing	-1.55	9.08	-1.98	1.12	0.26	0.69	0.49	0.26	0.24
prince-toronto	-1.05	27.75	53.26	1.04	0.14	0.09	0.36	0.18	0.18
gurlitt	-	25.5	-1.95	-	0.14	0.68	-	0.31	0.25

Table 6.4: Properties of the datasets for each of the events and each of the tasks, where S - Stance, V - Veracity, and D - Detection.

6.6.3 Analysis of data properties

Alonso and Plank (2017) link the gains/losses in performance from multi-task learning with information-theoretical metrics, properties of the label distributions: entropy (indicating the amount of uncertainty in the distribution) and kurtosis (indicating the skewness of the distribution). They have shown that the multi-task learning setup works best, when auxiliary tasks have label distributions with lower kurtosis and relatively high entropy. Table 6.4 shows the kurtosis and entropy properties of the label distribution for each of the events in the dataset, as well as token-type ratio (TTR). Each of the events has very different properties; there is a strong difference in properties between larger events (top) and smaller ones (bottom). Smaller events tend to have more extreme values, which may explain that their addition to the evaluation adds further complexity to the task of rumour verification. In line with findings of Alonso and Plank (2017), in our case both auxiliary tasks have on average lower kurtosis than the main task. The stance classification dataset has on average higher entropy than the rumour detection dataset.

Whereas Alonso and Plank (2017) were considering low-level linguistically related tasks as auxiliary tasks, such as part-of-speech tagging, we are working with higher level theme-related tasks. Therefore, it is interesting that we still observe similar trends when looking at the same data properties.

6.6.4 Incorporating additional features into multitask learning model

Further, we have performed experiments of incorporating various feature sets (as defined in section 5.4) into the multitask learning model that combines all three tasks (MTL3). We perform the experiments on the 5 largest events in the PHEME

MTL3 CV5	Accuracy	Macro-F	True	False	Unverified
text	**0.492**	**0.396**	0.647	0.211	0.330
text+attachments	0.353	0.318	0.507	0.278	0.169
text+interactions	0.463	0.381	0.593	0.139	0.409
text+lexicon	0.302	0.284	0.394	0.273	0.184
text+punctuation	0.414	0.391	0.497	**0.295**	**0.382**
text+user	0.471	0.256	**0.654**	0	0.114
text+tree	0.424	0.344	0.592	0.269	0.173
all features	0.366	0.218	0.541	0	0.114

Table 6.5: Results of adding extra features to the MTL3 model with 5 largest events from PHEME dataset. Results are presented in terms of overall accuracy and macro-f score as well as per-class f scores.

dataset. Table 6.5 present the results in terms of overall accuracy and macro-f score as well as per-class f scores. None of the additional features help improve the overall performance on veracity classification task over the performance of the model using textual features only. The per-class performance break down shows that all of the models perform best on the majority class *true*, however, interestingly, the best performance on each of the classes is achieved by models incorporating extra features. The identification of *true* class benefits from user features, while identification of *false* and *unverified* peaks when using punctuation features. Also, different sets of features affect performance on *false* and *unverified* classes to have different balance. In most cases (five out of eight experiments) models perform better on *unverified* class comparing to *false*, however incorporating attachments, lexicon and conversation tree features leads to improvements in performance on of the *false* class. These features would be preferred if one was mostly interested in identifying *false* without concern for other classes.

6.6.5 Performance of multitask learning model on Emergent dataset

The Emergent dataset contains annotations for two tasks: claim veracity classification and stance classification of articles discussing the claim towards the truthfulness of the claim. The instances in the dataset (as described in chapter 3) are article headlines/summaries written by fact-checkers rather than social media posts that we considered earlier in this chapter (even though some are based on materials from social media). The responses are not linked through the conversation, but via a topic, a claim in question. Rather than being arranged in branches, they can be arranged in a timeline according to the publication date. As we discussed previously the benefit of a multitask learning approach can depend on the dataset properties

Veracity	Majority	SVM	branch-LSTM	MTL2
MacroF	0.196	0.366	**0.503**	0.432
Accuracy	0.416	0.417	**0.52**	0.453

Table 6.6: Performance comparison for different models for veracity classification on Emergent dataset

Stance	Majority	SVM	branch-LSTM	MTL2
MacroF	0.213	0.496	0.229	**0.414**
Accuracy	0.469	0.537	0.487	**0.598**

Table 6.7: Performance comparison for different models for stance classification on Emergent dataset

such as richness of the language and label distribution, thus we investigate how a multitask approach will generalise to this case.

The authors of the Emergent dataset (Ferreira and Vlachos, 2016) provide the split into training and testing subsets. However as no development set was provided we chose to split the dataset into a stratified 5-folds for cross-validation to keep the class balance across the folds. For hyper-parameter tuning we use 3 folds to train the model, one fold to use as development and one is reserved for testing. We find a set of hyper-parameters that leads to best performance on the development set. Then we perform 5-fold cross-validation using that set of hyper-parameters. As we have chosen to perform cross-validation and defined our own split for this dataset, we are not performing the comparison with previous works as it would be unfair to compare against models developed under different evaluation set up. We represent claim and headlines as the average of word2vec word embeddings.

We evaluate performance in terms of macro-averaged F score for both tasks. We compare a multitask learning approach (*MTL2*) with single-task learning models. First we show a simple majority baseline to highlight some class imbalance present in the data. Then, a linear SVM that takes a claim as an input for a veracity classification task and a representation of an individual article for stance classification task. We also experiment with *branch-LSTM* model. However as the Emergent dataset does not contain tree-structured conversations, we use timelines, sequences of statements starting with a claim followed by responses arranged according to the time of their publication, as input. Tables 6.6 and 6.7 show the performance of all of the baselines and the multitask learning model. Interestingly, multitask learning model, while beating the majority and SVM baselines, did not improve performance on the veracity classification over the *branch-LSTM* model.

	Kurtosis		Entropy		TTR
folds	Stance	Veracity	Stance	Veracity	
0	-0.047	-1.218	0.876	1.063	0.550
1	-1.11	-1.21	0.629	1.063	0.559
2	-0.955	-1.23	1.018	1.064	0.593
3	-0.583	-1.182	0.943	1.056	0.599
4	-0.576	-1.182	0.942	1.055	0.532

Table 6.8: Properties of the datasets for each of the cross-validation folds and each of the tasks, where S - Stance and V - Veracity

However it reached highest accuracy for the stance classification task.

Table 6.8 shows the dataset properties: kurtosis and entropy of a label distribution as well as token-type ratio. As we have performed stratified cross-validation the properties of the folds are very similar. Previous studies pointed that tasks with lower kurtosis and higher entropy levels usually benefit from multitask learning approach. In the Emergent dataset the kurtosis is low for both of the tasks and entropy is consistently higher for veracity task than stance. Therefore the outcomes we have lead us to consider that there are more factors that lead to success of MTL appraoch. For example, TTR of the folds in the Emergent dataset is drastically higher than in the PHEME dataset. Another avenue for exploration could be, as suggested in Bingel and Søgaard (2017), to study the learning curves for the models tackling each task, which we leave to future work.

6.7 Conclusions and future work

We have proposed a rumour verification model that achieves improved performance for veracity classification by leveraging task relatedness with auxiliary tasks, specifically rumour detection and stance classification, through a multi-task learning approach. We have compared single task learning approaches with the proposed multitask learning approaches that combine the verification classifier with the stance and rumour detection classifiers individually, as well as with both the rumour detection system and the stance classifier. Our results show that the joint learning of two tasks from the verification pipeline outperforms a single-learning approach to rumour verification. The combination of all three tasks leads to further performance improvements. We have also investigated the link between the properties of the label distribution in the dataset and the outcomes of our multi-task learning models. Our results support findings from previous research (Alonso and Plank, 2017) in other tasks. The experiments on the Emergent dataset has shown that multitask

learning setup is not a guarantee of performance improvements on all of the tasks and the reasons as well as the cues for its success are yet to be explored.

Further, we extended the experiments with multi-task learning models on the PHEME dataset by incorporating extra features and evaluating their effect on the models performance. We did not see improvements over the model using just textual features in terms of overall performance. However different feature sets lead to improvements in performance of the model on certain classes, which would be important to study further and use if one of the classes is prioritised.

Future work could include investigating whether further improvements on main and auxiliary tasks are possible with multi-task learning by: adapting the training schedule to account for different dataset sizes, such that none of the tasks dominates the model; by incorporating the hierarchy between tasks into the model. Whether the effect of adding extra features on different tasks could depend on the stage on which they are incorporated, in private or shared layers.

CHAPTER 7

Incorporating Uncertainty Estimation into Rumour Verification

7.1 Introduction

There are many challenges in rumour verification: information about real-world events such as natural disasters and public crises appears in a fragmented, piecewise manner; often there can be an adversarial force generating disinformation for political or economical gains. The spread of misinformation at a time of crisis can have a particularly harmful impact, its correct resolution is crucial, and incorrectly approving false information can magnify the scale of the harmful effects. It is therefore highly desirable that an automated system in aid of rumour verification can inform a human fact checker of its level of uncertainty.

Deep learning models are currently the state-of-the-art in many NLP tasks, including rumour detection (Ma et al., 2018a), the task of identifying candidate rumours, and rumour verification (Kumar and Carley, 2019), where the goal is to resolve the veracity of a rumour. Latent features and large parameter spaces of deep learning models make it hard to interpret a model's decisions. Increasingly researchers are investigating ways to understand model predictions, such as by using neural attention (Vaswani et al., 2017) and studying adversarial examples (Yuan et al., 2019). Another way to gain insights into a model's decisions is via estimating its uncertainty. Understanding what a model does not know can help us determine when we can trust its output and at which stage information needs to be passed on to a human. Moreover, uncertainty estimates can provide insights about the data at hand and the task itself (Kendall and Gal, 2017).

In this chapter, rather than purely focusing on the performance of a rumour verification model, we estimate its predictive uncertainty to gain understanding of a model's decisions, thus addressing research question RQ7 as outlined in chapter 1. We consider two types of uncertainty: data uncertainty (aleatoric) and model uncertainty (epistemic). The approach we adopt requires minimal changes to a given model and is relatively computationally inexpensive, thus making it possible to apply to various architectures.
We make the following contributions:

- We are the first to apply methods for uncertainty estimation to the problem of rumour verification. We show that removing instances with high uncertainty filters out many incorrect predictions, gaining performance improvement in the rest of the dataset.
- We propose a supervised method for instance removal that combines both aleatoric and epistemic uncertainty and outperforms an unsupervised approach.
- We propose a way to analyse uncertainty patterns as a rumour unfolds in time. We make use of this to study the relation between the stance expressed in response tweets and fluctuation in uncertainty at the time step following a response.
- We explore the relationship between uncertainty estimates and class labels.
- We analyse parameters affecting uncertainty estimates and hence the amount of potential performance increase.

Adopting methods for uncertainty evaluation in rumour resolution systems will lead towards more interpretable and generalisable models.

7.2 Related work

There is a growing body of literature which aims to estimate predictive uncertainty of deep neural networks (DNNs) (Gal and Ghahramani, 2016b; Lakshminarayanan et al., 2017; Malinin and Gales, 2018). Gal and Ghahramani (2016b) have shown that application of Monte-Carlo (MC) Dropout at testing time can be used to derive an uncertainty estimate for a DNN. Lakshminarayanan et al. (2017) estimate model uncertainty by using a set of predictions from an ensemble of DNNs, while Malinin and Gales (2018) propose a specialised framework, Prior Networks, for modelling predictive uncertainty. Here we focus on the dropout method proposed by Gal and Ghahramani (2016b) as it is computationally inexpensive, relatively simple and does not interfere with model training.

Within NLP Xiao and Wang (2018) have used aleatoric (Kendall and Gal, 2017) and epistemic (Gal and Ghahramani, 2016b) uncertainty estimates for Sentiment analysis and Named Entity Recognition. Dong et al. (2018) used a modification of Gal and Ghahramani (2016b) method to output confidence scores for Neural Semantic Parsing. Rumour Verification is a task where levels of certainty play a crucial role because of the potentially high impact of erroneous decisions. Moreover, unlike other tasks, it is a time-sensitive problem: as new information comes to light the level of certainty is expected to change giving insights into a models predictions. We therefore explore the dynamics of uncertainty as a discussion unfolds in section 7.6.3.

Existing works on automated rumour verification aim to improve performance of supervised learning algorithms that classify claims, leveraging linguistic cues, network- and user-related features, propagation patterns, support among responses and conversation structure (Derczynski et al., 2017; Gorrell et al., 2019). Due to the nature of the task, as each rumour can be considered as a new domain, existing models struggle with generalisability. Here we are utilising model-agnostic methods of uncertainty estimation that would provide performance improvements and insight on the working of the models to inspire further development.

Note that data and model uncertainty should not be confused with uncertainty expressed by a user in a post. Automatically identifying levels of uncertainty expressed in text is a challenging NLP task (Jean et al., 2016; Vincze, 2015), which could be complementary to predictive uncertainty in the case of rumour verification.

7.3 Data

In this chapter we use the 9 events in the PHEME dataset annotated for verification, and more recent Twitter 15 and Twitter 16 datasets. Detailed description of all datasets can be found in Chapter 3. The Twitter 15 and Twitter 16 datasets were chosen as they contain the data in a similar format of tree-like conversations annotated with rumour veracity labels. They are relatively large, compared to RumourEval 2017, 2019. Importantly, the label distribution and cross validation split properties are very different between PHEME and Twitter 15, Twitter 16, hence it is interesting to observe the change in model performance due to those. As highlighted by the findings in chapter 5 it is crucial to test veracity classification models using multiple different datasets to find generalisable approach.

7.4 Methodology

7.4.1 Rumour verification model

For our experiments we utilise the *branch-LSTM* model as described in section 5.3.3. To process a conversation discussing a rumour while preserving some of the structural relations between the tweets, the conversation is split into branches of linear sequences of tweets as shown on figure 4.2. Branches are then used as training instances for a model consisting of an LSTM layer followed by several ReLU layers and a softmax layer that predicts class probabilities. Here we use outputs from the final time steps (see figure 5.1). Given a training instance, branch of tweets z_i, $i \in [1, .., \sum_{k=1}^{K} M_k]$, where K is the number of conversations in the dataset and M_k is the number of branches in the conversation k, and the label y_i, represented as one-hot vector of size C, where C is the number of classes, the loss function l_1 (categorical cross entropy) is calculated as follows:

$$u_i = f(z_i) \tag{7.1}$$

$$v_i = W_v u_i + b_v \tag{7.2}$$

$$p_i = \text{softmax}(v_i) = \frac{e^{v_i}}{\sum_{c=1}^{C} e^{v_i^c}} \tag{7.3}$$

$$l_1 = -\frac{1}{N} \sum_{n=1}^{N} \sum_{c=1}^{C} y_i^c \log p_i^c, \tag{7.4}$$

where u_i is an intermediate output of layers prior to the softmax layer, v_i is logits, and p_i are predicted class probabilities for a training instance z_i. To obtain predictions per conversation tree we average class probabilities for each of the branches from the tree. In chapters 5 and 6 we have used majority voting over per-branch predictions to aggregate predictions per-tree. Here we experimented with both majority voting and averaging of output of softmax layer and found no difference in performance in most cases (in some experiments there were minor differences, where one or two instances would be classified differently by the two approaches). The model is trained using categorical cross entropy loss. Tweets are represented as the average of the corresponding word2vec word embeddings, pre-trained on the Google News dataset (300d) (Mikolov et al., 2013a).

7.4.2 Uncertainty estimation

We consider two types of uncertainty (Kendall and Gal, 2017): data uncertainty (aleatoric) and model uncertainty (epistemic). Data uncertainty is normally associated with properties of the data, such as imperfections in the measurements. Model uncertainty on the other hand comes from model parameters and can be explained away given enough (i.e. an infinite amount of) data.

7.4.2.1 Data uncertainty

We assume aleatoric uncertainty to be a function of the data that can be learned along with the model (Kendall and Gal, 2017). Conceptually, this input-dependent uncertainty should be high when it is hard to predict the output given a certain input.

In order to estimate aleatoric uncertainty associated with input instances, we add an extra output to our model that represents variance σ. We then incorporate σ into the loss function according to Kendall and Gal (2017), in the following way.

$$\sigma_i = \text{softplus}(W_\sigma u_i + b_\sigma) = ln(1 + e^{W_\sigma u_i + b_\sigma}) \tag{7.5}$$

Here we assume that predictions come from normal distribution with mean v and variance σ. We sample v, distorted by Gaussian noise, T times, put each through a softmax layer and pass to a standard categorical cross entropy loss function to obtain a mean over losses for all T samples.

$$d_{t,i} = v_i + \sqrt{\sigma_i} * \epsilon, \quad \epsilon \sim N(0, 1) \tag{7.6}$$

$$l_2 = -\frac{1}{N} \sum_{n=1}^{N} \frac{1}{T} \sum_{t=1}^{T} \sum_{c=1}^{C} y_i^c \log(\text{softmax}(d_{t,i})^c) \tag{7.7}$$

Here $l = w_1 l_1 + w_2 l_2$ is the total loss.

If the original prediction u was incorrect, we would need a high σ to have varied samples away from it and hence lower the loss. In the opposite case, σ should be small such that all samples yield a similar result, thus minimising the loss function. σ is chosen as the unbound variance in logit space, which after the model is trained approximates variance caused by the inputs. This method can be applied to a wide range of models, but since it changes the loss function, it may affect a model's performance.

7.4.2.2 Model uncertainty

To obtain epistemic uncertainty we use a well-established and widely accepted approach proposed by Gal and Ghahramani (2016b). This approach allows estimating uncertainty about a model's predictions by applying dropout at testing time and sampling from the approximate posterior. This approach requires no changes to the model, does not affect performance, and is relatively computationally inexpensive. We apply dropout at testing time N times and obtain N predictions. We evaluate the differences between them to obtain a single uncertainty value in the following ways:

Variation ratio Each of the sampled softmax predictions can be converted into an actual class label. We then define epistemic uncertainty as the proportion of cases which are not in the mode category (the label that appears most frequently).

$$v = 1 - N_m/N_{total},$$

where N_m is the number of cases belonging to the mode category (most frequent class). Thus the variation ratio is 0 when all of the sampled predictions agree, indicating low model uncertainty. The upper bound would differ depending on the number of cases, but will not reach 1.

Entropy Given an array of predictions, we average over them and then calculate predictive entropy as follows:

$$s = -\sum_i p_i \log p_i.$$

Variance Each prediction is a vector, the output of a softmax layer (entries in [0,1] which sum up to 1), of size equal to the number of classes. We calculate the variance across each dimension and then take the max value of variance as our uncertainty estimate.

7.4.3 Instance rejection

To evaluate the effectiveness of the above methods, we remove instances with high uncertainty values as they are likely to be incorrectly predicted, such instances then could be passed on to a human to make a judgement. This allows us to explore the trade-off between model performance and coverage of a dataset. We perform instance rejection in the following ways.

Unsupervised We remove portions of a dataset corresponding to instances with the highest uncertainty (separately for each type). We also consider the output of the softmax layer as a measure of a model's confidence and use the lowest softmax values obtained to perform unsupervised rejection. The removal of instances is applied to the testing set, as we perform experiments using cross-validation, each fold becomes testing once.

Supervised We train a supervised meta-classifier on features composed of uncertainty estimates (aleatoric, variance, entropy, variation ratio), the averaged softmax layer output and the model's prediction to decide whether an instance is correctly predicted. We reject instances classified as incorrect by this meta-model and evaluate performance on the rest. We compare two models for this task: Support Vector Machines (SVM) and Random Forest (RF). Supervised rejection allows us to leverage all forms of uncertainty together and also dictates the number of instances to remove.

Random We have compared the two instance rejection methods above against removing portions of the test set at random. The outcome of the rejection at random does not lead to consistent performance improvement.

7.4.4 Time-sensitive uncertainty estimates

Since rumour verification is a time-sensitive task, we have performed analysis of model uncertainty over time, as a rumour unfolds. As illustrated in figure 7.1 we have deconstructed the timeline of the development of a conversation tweet by tweet, starting with just the source tweet (initiating the rumour) and adding one response at a time. We have then obtained model predictions and associated uncertainties for each sub-tree. As the difference between each sub-tree is a single tweet, we can track the development of uncertainty alongside the development of a conversation, and the effect each added response has.

7.5 Experimental setup

7.5.1 Training and evaluation setup

We perform cross-validation on all of the datasets. When choosing parameters, we choose one of the folds within each dataset to become the development set: CharlieHebdo in PHEME (large fold with balanced labels) and fold 0 in Twitter 15

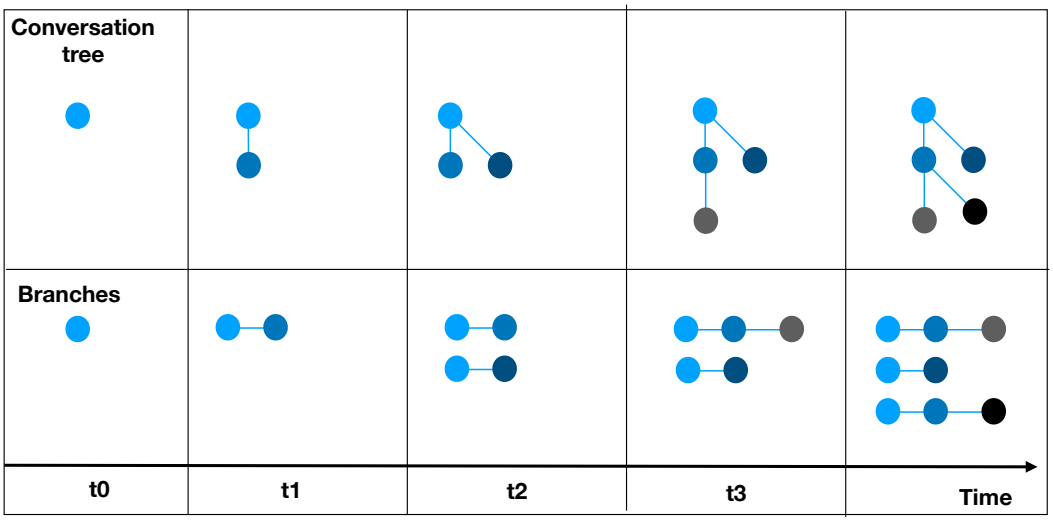


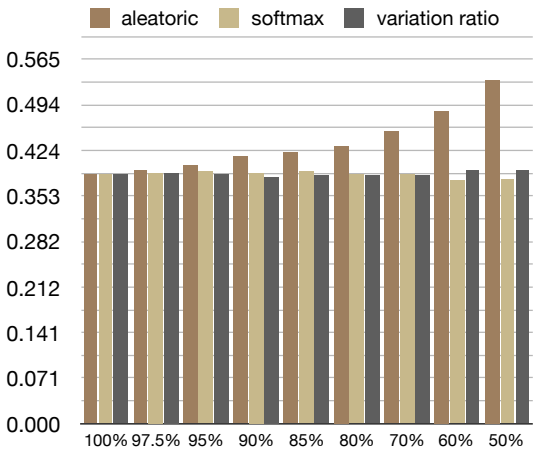
Figure 7.1: Development of a conversation tree over time and its decomposition into branches

and Twitter 16. We evaluate models using both accuracy and macro-averaged F score due to the class imbalance in the PHEME dataset.

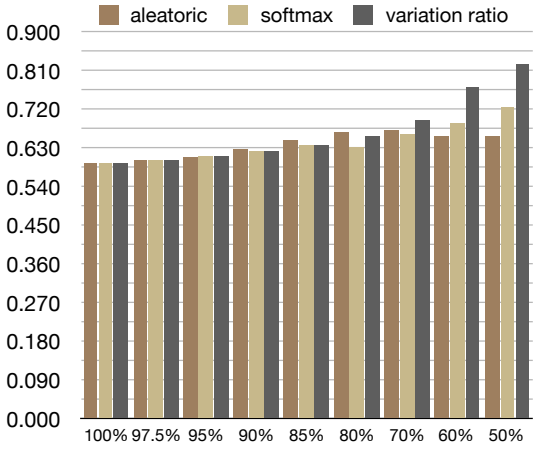
7.5.2 Development set for supervised rejection

To perform supervised rejection we need to train a meta-classifier on a subset of data that was not used for training the rumour verification model. Therefore in a separate set of experiments we exclude one of the folds (development set) from training of the verification model. We run cross-validation with one less fold and at each step obtain predictions and uncertainty estimates for both the test fold and the development set. We then use the predictions and uncertainty values predicted for the instances in the development set as training instances in our rejection meta-models, which we then evaluate on each of the corresponding test folds, thus obtaining the combined predictions for all of the folds in the dataset except for the development. This setup corresponds to results shown in table 7.7, as one of the folds was removed from training.

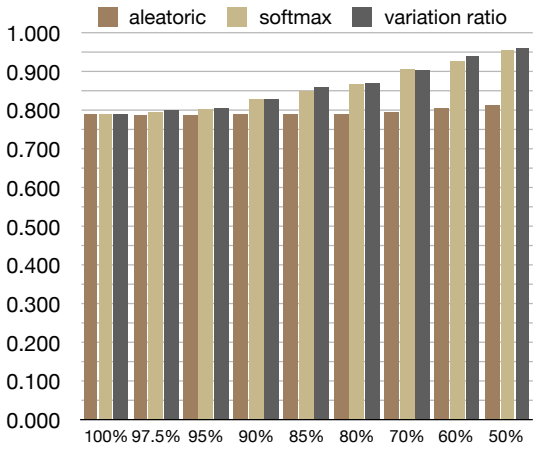
We perform the experiments in two different setups: (1) including development set in training and evaluation when testing unsupervised rejection (the results in figure 7.2) in order to obtain results comparable to the previous literature on the PHEME, Twitter 15 and Twitter 16 datasets and (2) excluding it for supervised rejection strategy as described above (the results in table 7.7) in order to evaluate this supervised rejection strategy and comapre it with unsipervised.



(a) PHEME



(b) Twitter 15



(c) Twitter 16

Figure 7.2: Unsupervised rejection of instances with the highest uncertainty and corresponding lowest confidence (softmax) values across 3 datasets. The Y-axis shows performance in terms of accuracy, on the X-axis the percentage of the remaining instances is shown.

7.6 Results

In this section we show the effect of using estimated uncertainties to perform instance rejection.

7.6.1 Unsupervised rejection

Figure 7.2 shows the effect of applying unsupervised rejection (as explained in section 7.4.3). Each plot shows model performance in terms of accuracy, where the first bar of each plot shows model performance with all instances present and the following bars show performance for the corresponding percentage of remaining instances. Figure 7.2 shows the effect of unsupervised rejection using aleatoric and epistemic uncertainty (calculated as variation ratio, see section 7.4.2.2), as well as the softmax class probabilities as a measure of confidence (1-uncertainty). We have also performed experiments with variance and entropy values and achieved similar results. Tables 7.1-7.3 present the results in terms of accuracy of unsupervised rejection of instances with the highest uncertainty and corresponding lowest confidence (softmax) values against random rejection of instances across 3 datasets: PHEME, Twitter 15, Twitter 16.

Initial performance using 100% of the data (Figure 7.2) on the PHEME dataset is markedly different to Twitter 15 and Twitter 16 due to the dataset and task-setup differences. On the Twitter 15 dataset *branch-LSTM* does not reach the state-of-the-art Tree-GRU (Ma et al., 2018a), however *branch-LSTM* outperforms Tree-GRU on the Twitter 16 dataset. On the PHEME dataset performance is comparable and slightly improved over the results in chapter 6.

In line with model performance, the effect of rejection using aleatoric and epistemic uncertainties is different for PHEME compared to Twitter 15 and Twitter 16. Figure 7.2 (a) shows that in PHEME greater improvement in accuracy comes from using aleatoric uncertainty, whereas for Twitter 15 (b) and Twitter 16 (c) there is very little improvement with aleatoric uncertainty compared to epistemic. We believe this is due to the nature of the datasets: folds in PHEME differ widely in size and class balance, resulting in higher/more varied data uncertainty values, in contrast with the very balanced datasets of Twitter 15 and Twitter 16. The effect of rejection using low values of softmax confidence is also positive and often similar to the effect of epistemic uncertainty as it is also estimating model's uncertainty. However softmax is outperformed by other types of uncertainty in most cases (figure 7.2).

We have also calculated macro-averaged f1-score and improvements after re-

%	# removed instances	Random	Aleatoric	Entropy	Softmax	Variance	Variation ratio
100%	0	0.385	0.385	0.385	0.385	0.385	0.385
97.5%	60	0.384	0.391	0.388	0.386	0.387	0.386
95%	120	0.384	0.397	0.388	0.386	0.387	0.386
90%	240	0.382	0.412	0.385	0.387	0.387	0.387
85%	361	0.384	0.417	0.385	0.388	0.385	0.386
80%	481	0.381	0.427	0.385	0.388	0.385	0.387
70%	723	0.374	0.448	0.389	0.387	0.389	0.388
60%	964	0.370	0.481	0.387	0.389	0.396	0.394
50%	1205	0.376	**0.528**	0.389	0.386	0.392	0.391

Table 7.1: Performance (accuracy) after unsupervised rejection on PHEME dataset for all types of uncertainty.

%	# removed instances	Random	Aleatoric	Entropy	Softmax	Variance	Variation ratio
100.0%	0	0.591	0.591	0.591	0.591	0.591	0.591
97.5%	34	0.589	0.599	0.603	0.601	0.599	0.602
95.0%	68	0.593	0.61	0.609	0.609	0.609	0.609
90.0%	137	0.592	0.63	0.625	0.621	0.627	0.622
85.0%	206	0.597	0.647	0.637	0.634	0.646	0.634
80.0%	274	0.599	0.668	0.648	0.63	0.665	0.657
70.0%	412	0.577	0.642	0.669	0.66	0.718	0.699
60.0%	549	0.596	0.64	0.679	0.684	0.77	0.765
50.0%	687	0.598	0.649	0.677	0.723	0.817	**0.821**

Table 7.2: Performance (accuracy) after unsupervised rejection on Twitter 15 dataset for all types of uncertainty.

%	# removed instances	Random	Aleatoric	Entropy	Softmax	Variance	Variation ratio
100.0%	0	0.788	0.788	0.788	0.788	0.788	0.788
97.5%	18	0.789	0.784	0.798	0.795	0.794	0.796
95.0%	36	0.787	0.783	0.808	0.8	0.805	0.805
90.0%	73	0.787	0.787	0.837	0.828	0.828	0.829
85.0%	110	0.786	0.787	0.856	0.848	0.85	0.856
80.0%	146	0.789	0.789	0.881	0.864	0.868	0.869
70.0%	220	0.794	0.794	0.905	0.905	0.907	0.901
60.0%	294	0.787	0.803	0.939	0.925	0.937	0.937
50.0%	367	0.78	0.81	0.954	0.951	**0.957**	**0.957**

Table 7.3: Performance (accuracy) after unsupervised rejection on Twitter 16 dataset for all types of uncertainty.

%	# removed instances	Random	Aleatoric	Entropy	Softmax	Variance	Variation ratio
100%	0	0.311	0.311	0.311	0.311	0.311	0.311
97.5	60	0.309	0.312	0.310	0.309	0.310	0.309
95	120	0.310	**0.314**	0.307	0.307	0.310	0.309
90	240	0.308	**0.315**	0.302	0.303	0.304	0.305
85	361	0.310	0.293	0.299	0.304	0.299	0.302
80	481	0.311	0.293	0.296	0.300	0.296	0.298
70	723	0.306	0.300	0.297	0.294	0.294	0.296
60	964	0.304	0.296	0.291	0.293	0.295	0.295
50	1205	0.306	0.287	0.291	0.289	0.288	0.290

Table 7.4: Performance (macro F) after unsupervised rejection on PHEME dataset for all types of uncertainty

%	# removed instances	Random	Aleatoric	Entropy	Softmax	Variance	Variation ratio
100%	0	0.597	0.597	0.597	0.597	0.597	0.597
97.5	34	0.598	0.603	0.609	0.607	0.603	0.608
95	68	0.598	0.614	0.615	0.614	0.614	0.615
90	137	0.597	0.631	0.630	0.626	0.629	0.627
85	206	0.598	0.645	0.642	0.640	0.647	0.639
80	274	0.601	0.663	0.653	0.635	0.661	0.659
70	412	0.59	0.637	0.675	0.665	0.711	0.692
60	549	0.605	0.626	0.686	0.689	0.756	0.751
50	687	0.608	0.639	0.685	0.722	0.799	**0.804**

Table 7.5: Performance (macro F) after unsupervised rejection on Twitter 15 dataset for all types of uncertainty

%	# removed instances	Random	Aleatoric	Entropy	Softmax	Variance	Variation ratio
100%	0	0.787	0.787	0.787	0.787	0.787	0.787
97.5	18	0.788	0.784	0.796	0.794	0.792	0.795
95	36	0.785	0.783	0.806	0.798	0.803	0.804
90	73	0.786	0.788	0.834	0.826	0.825	0.827
85	110	0.784	0.787	0.853	0.845	0.847	0.853
80	146	0.788	0.789	0.878	0.860	0.864	0.865
70	220	0.793	0.794	0.900	0.898	0.903	0.896
60	294	0.783	0.803	0.930	0.913	0.928	0.927
50	367	0.776	0.808	0.940	0.930	**0.944**	**0.944**

Table 7.6: Performance (macro F) after unsupervised rejection on Twitter 16 dataset for all types of uncertainty

jection in terms of it (tables 7.4-7.6). For Twitter 15 and Twitter 16 values of macro-F score and their improvement are very similar to accuracy. However for PHEME dataset we do not observe strong improvement of macro-F score when performing rejection. This is likely due to the class imbalance and model assigning high uncertainty to the instances from minority class that it struggles to identify, thus improving overall performance and still having poor performance on the minority class. In all cases random rejection does not lead to consistent performance improvements, and hence, is outperformed by (un)certainty-based rejection.

As discussed above, removing instances using uncertainty estimates leads to higher performance as higher levels of uncertainty indicate the incorrectly predicted instances. Using epistemic uncertainty is more efficient on Twitter 15 and 16 dataset, while aleatoric is better for PHEME dataset. Softmax-based rejection also leads to improvements, but is outperformed by either aleatoric or epistemic estimates depending on the dataset.

	All instances		Classifier	N removed	Supervised rejection		Unsupervised rejection					
							aleatoric		epistemic (variation ratio)		softmax	
	Accuracy	Macro F			Accuracy	Macro F	Accuracy	Macro F	Accuracy	Macro F	Accuracy	Macro F
PHEME	0.278	0.225	SVM	1057	**0.399**	0.196	0.306	0.216	0.35	0.235	0.332	0.239
			RF	1179	**0.378**	0.235	0.311	0.217	0.346	0.227	0.329	0.236
Twitter 15	0.671	0.67	SVM	402	**0.806**	**0.801**	0.656	0.632	0.801	0.795	0.794	0.788
			RF	504	**0.834**	**0.829**	0.662	0.624	0.836	0.828	0.818	0.811
Twitter 16	0.755	0.756	SVM	184	**0.895**	**0.893**	0.751	0.744	0.885	0.878	0.878	0.868
			RF	197	**0.897**	**0.892**	0.755	0.747	0.887	0.878	0.884	0.873

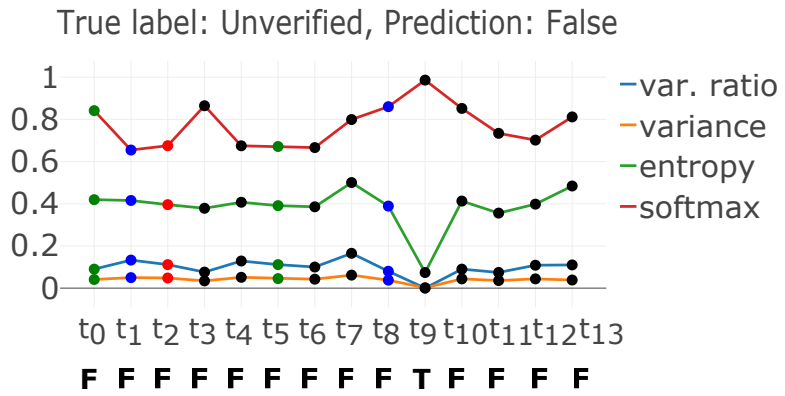
Table 7.7: Effects of rejecting instances using supervised and unsupervised methods on model performance across datasets, in terms of both accuracy and macro-F score. Performance values were obtained in a separate set of experiments, by removing one of the folds from the training set, as supervised models needed an extra development set to be trained on.

7.6.2 Supervised rejection

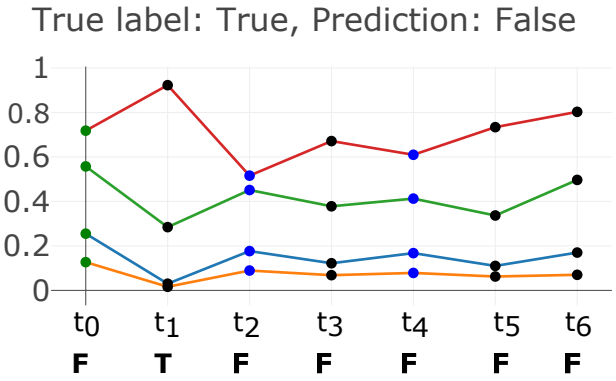
Table 7.7 shows the comparison of two models for supervised rejection versus unsupervised rejection of the same number of instances for all three datasets. Note that performance value in Table 7.7 differs from that in figure 7.2 as this was obtained in a separate set of experiments (as described in section 7.5). Having less training data harmed performance on PHEME and Twitter 16. Table 7.7 shows that using supervised rejection is better than unsupervised in terms of accuracy scores for all datasets and also in terms of macro-F scores for the Twitter 15 and Twitter 16 datasets. We believe that the reason the same effect on macro-F score is not observed in PHEME is the class imbalance in this dataset. As we are removing the same amount of instances using both supervised and unsupervised approaches, we are comparing their effectiveness, i.e. higher performance of supervised method means that it removes less correct predictions than unsupervised. Comparing the two methods, SVM and RF, for supervised rejection we observe that RF leads to a larger amount of instances being removed, achieving higher performance than SVM. However, the difference in performance between the two is very small. Another benefit of using a supervised model for instance rejection is that it can be further tuned, e.g., by varying the threshold boundary to prioritise high precision over recall. The precision value of this meta-classifier is the same as the accuracy of the predictions obtained after the rejection procedure.

7.6.3 Timeline analysis

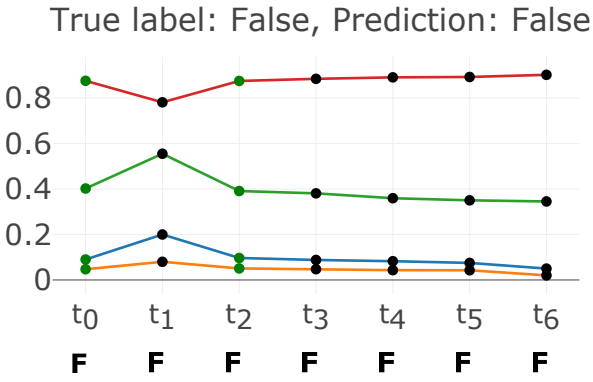
Figures 7.4 and 7.3 show examples of timelines of changes in predictions and uncertainty levels over time. Figure 7.4 shows all types of epistemic uncertainty: variation ratio (blue), entropy (green), variance (orange) as well as softmax confidence (red); while figure 7.3 shows aleatoric uncertainty of the conversations corresponding to the above plots separately, as values are on a different scale. Each of the nodes is labelled with its predicted stance label: green – supporting, red – denying, blue – questioning and black – commenting. As only a part of the PHEME dataset was annotated for stance (Derczynski et al., 2017), we used the *branch-LSTM* model proposed in chapter 5 trained on that part to obtain predicted stance labels for the rest of the PHEME dataset. There is no stance information for the Twitter 15 and Twitter 16 datasets, so this analysis is only available for the PHEME dataset. Note that we did not provide stance as a feature to train the veracity classifier: we assume that stance is an implicit feature within the tweets. Incorporating stance explicitly as a feature could provide stronger effect on uncertainty levels.



(a)

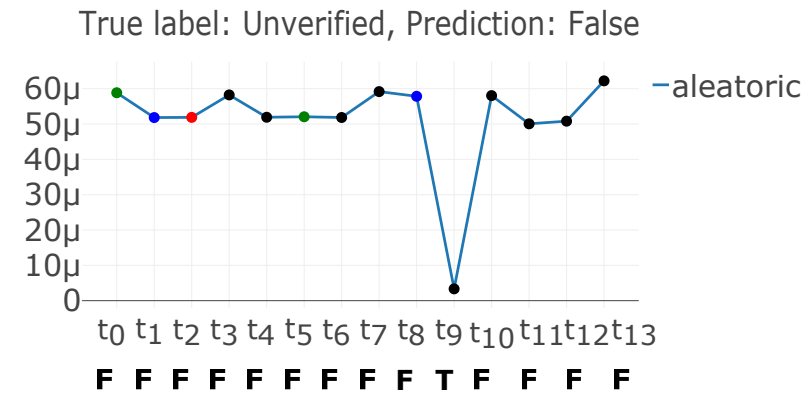


(b)

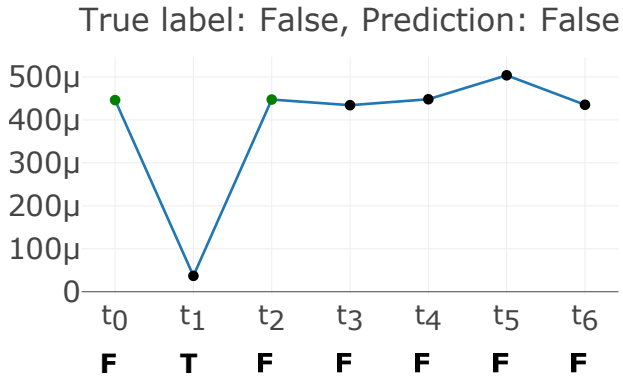


(c)

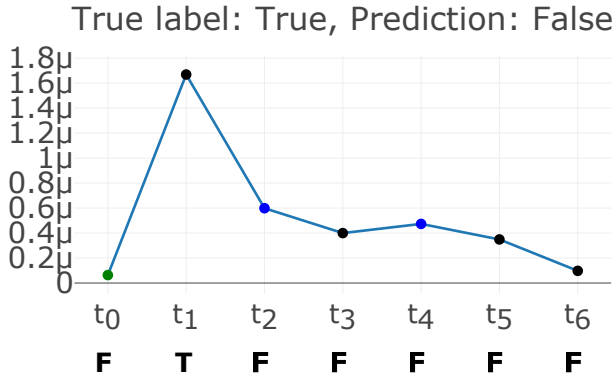
Figure 7.3: Examples of epistemic uncertainty development over time for three conversations discussing rumours from the PHEME dataset. Each of the nodes is labeled with its predicted stance label: green – supporting, red – denying, blue – questioning and black – commenting. Predictions are in bold at the bottom, where F – false, T – true, U – unverified.



(a)



(b)



(c)

Figure 7.4: Examples of aleatoric uncertainty development over time for three conversations discussing rumours from the PHEME dataset. Each of the nodes is labeled with its predicted stance label: green – supporting, red – denying, blue – questioning and black – commenting. Predictions are in bold at the bottom, where F – false, T – true, U – unverified.

One could expect to see uncertainty decreasing over time as more information about a rumour becomes available (we can see this effect only very weakly on subplot figure 7.4(b), showing a correctly predicted *false* rumour). However, not all responses are equally relevant and also the stance of new posts varies, therefore the uncertainty levels also change. Interestingly, the *true* rumour on subplot figure 7.4(a) (incorrectly predicted as *false* during the final time steps) had low uncertainty at step 2 and was predicting a correct label. However, the model appears to have been confused by further discussion resulting in an incorrect prediction with higher uncertainty levels. The analysis of uncertainty as a rumour unfolds can be used not only to analyse the effect of users' stance but also to study other properties of rumour spread. Furthermore, we can use the timelines of uncertainty measurements in order to only allow predictions at the time steps with lowest uncertainty, which may lead to performance improvements. Indicatively, in experiments with the PHEME dataset accuracy grew from 0.385 to 0.395 when using variation ratio and to 0.398 when using aleatoric uncertainty estimates.

When analysing the relation between uncertainty and the conversation size, we observed that for the confidence levels represented by the output of the softmax layer, conversations with a larger amount of tweets had higher uncertainty. However, for aleatoric and epistemic estimates we do not observe a strong trend of uncertainty increase with the size of the conversation (see box plots in supplementary material 3), which would indicate that these types of uncertainty are more robust in this respect. Higher levels of uncertainty associated with longer conversations may be due to the fact that responses became less informative and/or conversation changed topic. They may also be stemming from a weakness in model architecture in terms of its ability to process long sequences.

7.6.4 Uncertainty and conversation size

We have analysed how the size of the conversations affects uncertainty values. Figure 7.5 shows boxplots of uncertainty values of the conversations in PHEME, Twitter 15 and Twitter 16 datasets grouped by the number of tweets in each of them for aleatoric and epistemic uncertainty estimates as well as confidence levels (softmax). The conversations were grouped into equal sized bins, while resulting ranges of number of tweets are shown along the x-axis. We observe that for the confidence levels represented by the output of the softmax layer (figure 7.5 (g-i)), conversations with a larger amount of tweets score lower values i.e., they have higher uncertainty. However for aleatoric and epistemic estimates (figure 7.5 (a-f)) we do not observe a strong trend of uncertainty increase with the size of the conversation, so they seem

	True	False	Unverified	Non-Rumour
PHEME	0.569	0.198	0.163	-
Twitter 15	0.679	0.618	0.608	0.503
Twitter 16	0.88	0.729	0.755	0.739

Table 7.8: Per-class f1-scores of branch-LSTM model on each of the datasets.

to be more robust in this respect.

7.6.5 Uncertainty and class labels

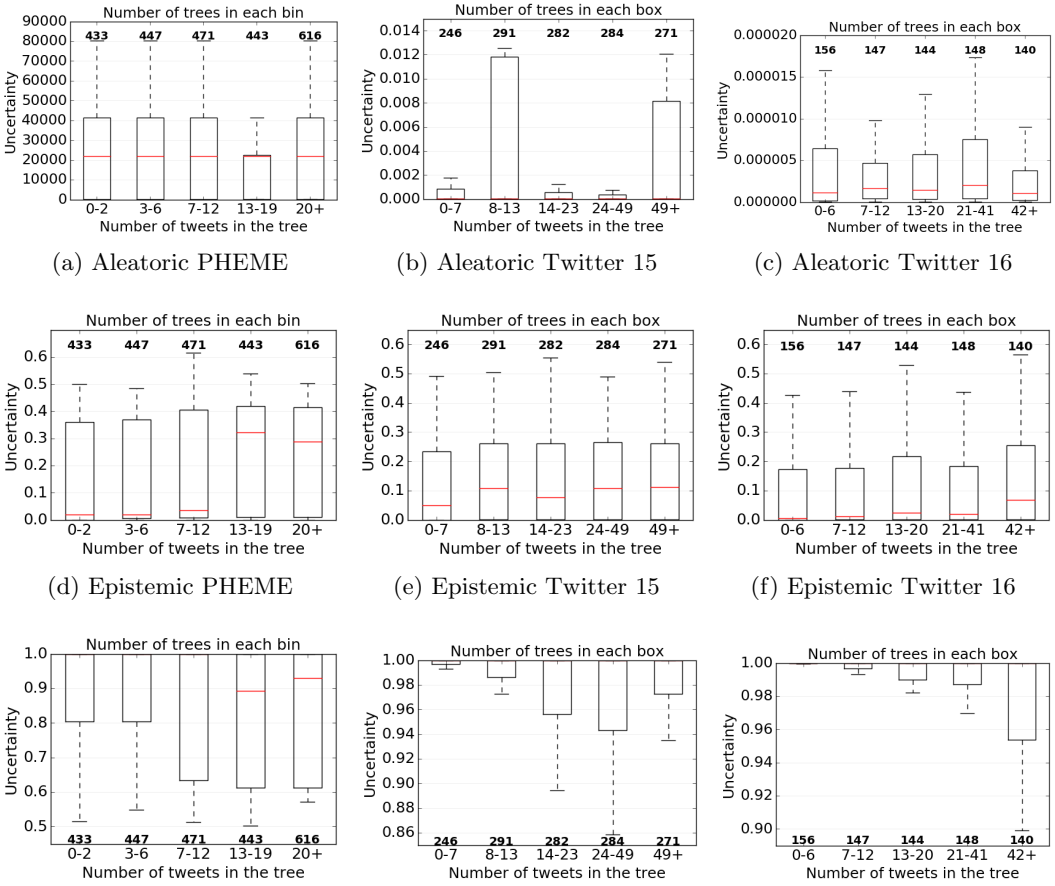
One of our research questions was whether higher uncertainty would be associated with a particular class label. Figure 7.6 shows boxplots of epistemic uncertainty values associated with each of the three classes in the PHEME dataset and each of the four classes in Twitter 15 and Twitter 16. Table 7.8 shows per-class model performance on the full datasets. In all datasets the *true* class has significantly lower levels of uncertainty (according to a Kruskal-Wallis test (Kruskal and Wallis, 1952) between the groups), while the uncertainties for *false* and *unverified* are higher than *true*. The difference between *false* and *unverified* is not statistically significant in any cases. Aleatoric uncertainty shows a similar pattern for the class labels. In Twitter 15 and Twitter 16 the Non-Rumour class has the highest uncertainty (and relatively lower f1 score). These outcomes are inline with findings in Kendall (2019) which showed an inverse relationship between uncertainty and class accuracy or class frequency.

7.7 Effect of parameters on uncertainty estimates

The methods we use for uncertainty estimates rely on a number of parameters.

For epistemic uncertainty the main parameter is the dropout probability as the method relies on applying dropout at testing time. Aleatoric uncertainty estimates depend on the number of times we perform sampling (T) and how much weight (w) the model places on optimising the loss function associated with uncertainty.

We have performed a small parameter sweep comparing the output of models with testing dropout in [0.1, 0.3, 0.5, 0.7], T in [10, 50] and w in [0.2, 0.5]. Plots on figure 7.7 show the effect of varying these parameters on unsupervised rejection outcomes in experiments on all datasets. In figure 7.7 the Y-axis shows accuracy and the X-axis the proportion of the dataset on which it is measured.

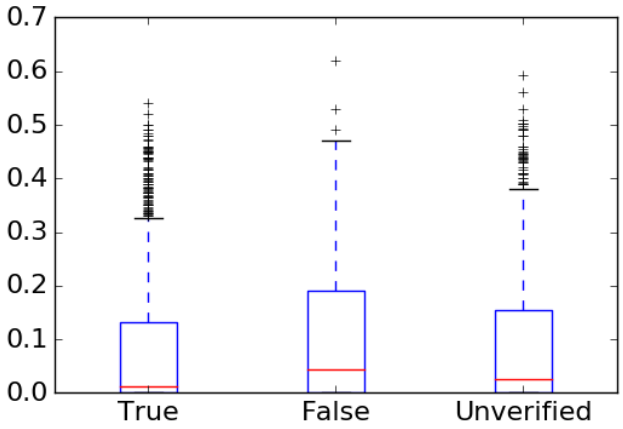


(a) Aleatoric PHEME (b) Aleatoric Twitter 15 (c) Aleatoric Twitter 16

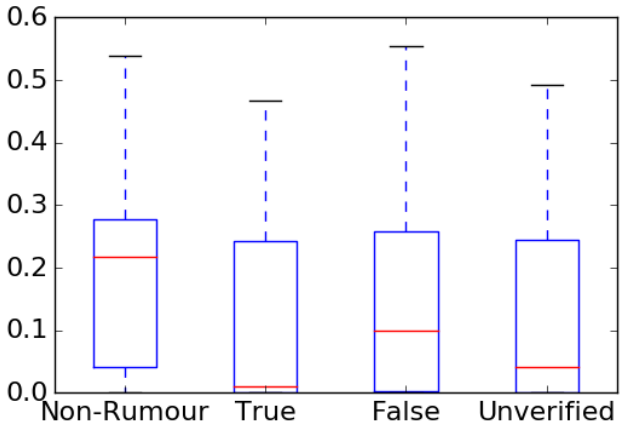
(d) Epistemic PHEME (e) Epistemic Twitter 15 (f) Epistemic Twitter 16

(g) Softmax PHEME (h) Softmax Twitter 15 (i) Softmax Twitter 16

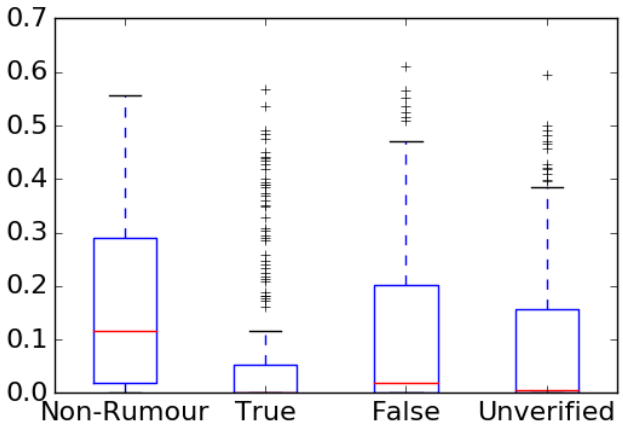
Figure 7.5: Boxplots showing uncertainty values grouped by the number of tweets in a conversation tree for 3 types of uncertainty estimates: aleatoric, epistemic, softmax. The Y-axis shows uncertainty (a-f) and confidence (g-i) values (a higher number indicates lower uncertainty). Numbers in bold show the number of conversations trees in each of the bins.



(a) Epistemic PHEME

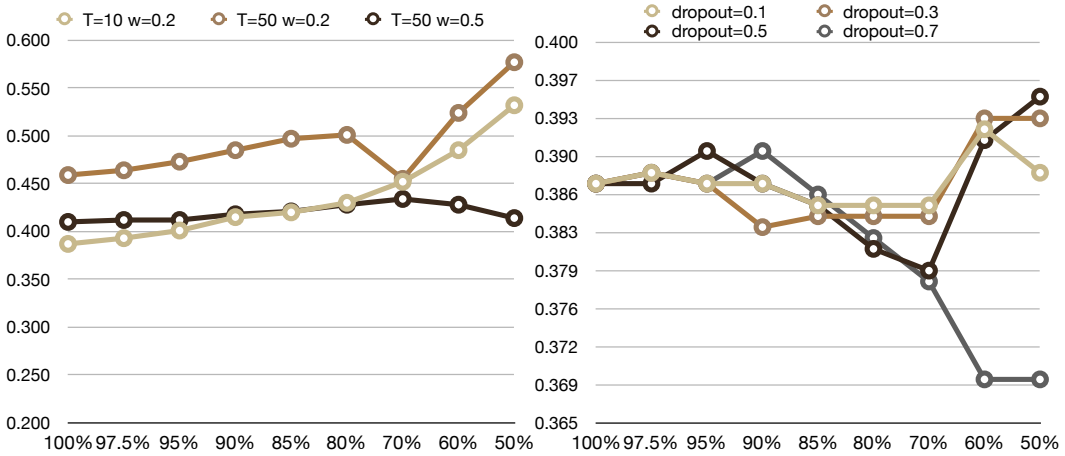


(b) Epistemic Twitter 15



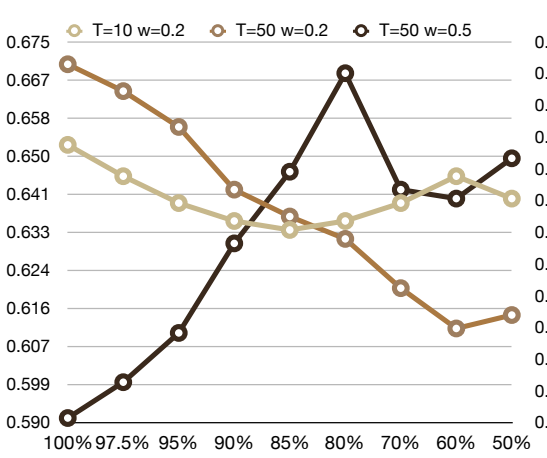
(c) Epistemic Twitter 16

Figure 7.6: Effect of class labels on uncertainty estimates.

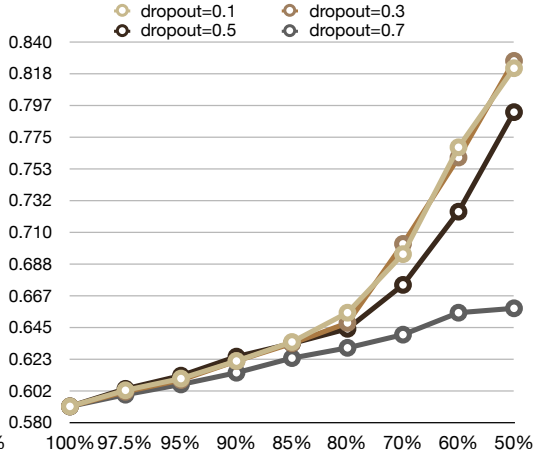


(a) Aleatoric PHEME

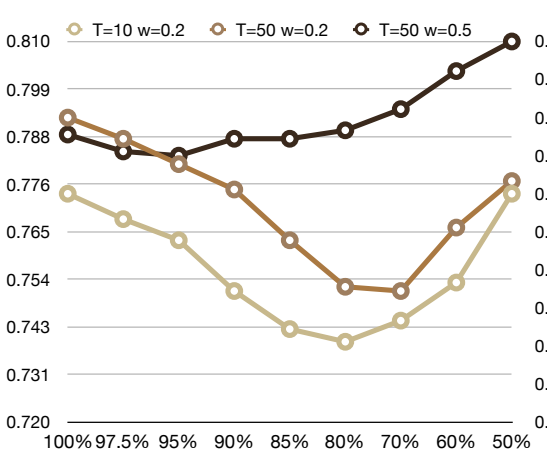
(b) Epistemic PHEME



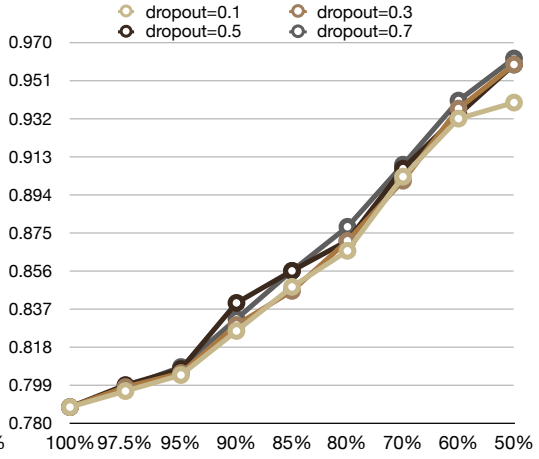
(c) Aleatoric Twitter 15



(d) Epistemic Twitter 15



(e) Aleatoric Twitter 16



(f) Epistemic Twitter 16

Figure 7.7: Effect of parameters on uncertainty estimates.

We see that the effect of parameters is dataset-dependent. The method for estimating aleatoric uncertainty affects a model's performance as it is incorporated in its loss function. By contrast estimating epistemic uncertainty using dropout at testing time does not have any effect on model performance.

On the plots for aleatoric uncertainty figure 7.7 (a,c,e) we see that changes in T and w strongly affect uncertainty estimates and the way they impact performance after unsupervised rejection. On the balanced Twitter 15 and Twitter 16 datasets aleatoric uncertainty for low T and w values does not help disambiguate between correct and incorrect instances very well and needs to be tuned by increasing their values. However, that may lead to deterioration of model performance, introducing a trade-off.

On the highly imbalanced PHEME dataset, aleatoric uncertainty estimates lead to improvements in performance for all parameter values, with the most increase observed when using a higher T and w=0.2. We have not tested values of T higher than 50, which could lead to further improvements. However it is likely there will be a maximum value after which we see no further improvements.

Varying the dropout rate during testing leads to changes in epistemic uncertainty estimates and their effect on performance using unsupervised rejection (figure 7.7 (b,d,f)). The performance gains are observed for all three datasets. Increasing the dropout parameter from 0.1 to 0.3 in all datasets, and up to 0.5 in the PHEME and Twitter 16 datasets, leads to further improvements compared to lower values. However further increase of dropout to 0.7 starts to damage performance on the PHEME and Twitter 15 datasets.

Varying the dropout rate (the dropout rate is changed for both stages, training and testing) leads to changes in epistemic uncertainty estimates. Higher dropout rate corresponds to a a higher increase in performance after rejection. We found that an increase in parameters increases the effect of uncertainty estimates, however have we tested higher values of those parameters we might have arrived at the decline.

7.8 Discussion

Data and model uncertainties can be included as part of the evaluation of any deep learning model without harming its performance. Even though data uncertainty estimation changes the loss function of a model, it often leads to improvements (Kendall and Gal, 2017).

When performing rejection in an unsupervised fashion we need to know when

to stop removing instances. Defining a threshold of uncertainty is not straightforward as uncertainty will be on a different scale for different datasets. Supervised rejection leverages all forms of uncertainty together and dictates the number of instances to remove. Thus to tune both methods availability of a development set is important.

While we are not focusing on user uncertainty here, in rumour verification linguistic markers of user uncertainty (such as words "may", "suggest", "possible") are associated with rumours. In the PHEME dataset such expressions often occur in unverified rumours, thus conversations containing them are easier to classify, and hence they are associated with lower predictive uncertainty.

7.9 Conclusions

We have presented a method for obtaining model and data uncertainty estimates on the task of rumour verification in Twitter conversations. We have demonstrated two ways in which uncertainty estimates can be leveraged to remove instances that are likely to be incorrectly predicted, so that making a decision concerning those instances can be prioritised by a human. We have also shown how uncertainty estimates can be used to interpret model decisions over time. We discussed the dataset-specific relation between uncertainty levels with conversation size and particular class labels. Our results indicate that the effect of data uncertainty and model uncertainty varies across datasets due to differences in their respective properties. The methods presented here can be selected based on knowledge of the properties of the data at hand, for example prioritising the use of aleatoric uncertainty estimates on imbalanced and heterogeneous datasets such as PHEME. For best results, one should use a combination of aleatoric and epistemic uncerainty estimates and tune the parameters of uncertainty estimation methods using a development set. Using uncertainty estimation methods can help identify which instances are hard for the model to classify, thus highlighting the areas where one should focus during model development.

Future work would include a comparison with other, more complex, methods for uncertainty estimation, incorporating uncertainty to affect model decisions over time, and further investigating links between uncertainty values and linguistic features of the input.

CHAPTER 8

Conclusions

8.1 Main findings

In the current section we summarise our main findings with respect to each of the research questions set up in chapter 1. We have grouped the research questions introduced there according to the two tasks we have considered in this thesis, rumour stance identification and rumour verification.

8.1.1 Rumour stance identification

RQ3 Can we automatically identify stance of users towards rumours from the posts in a conversation? We have studied several approaches to automated stance identification in Twitter conversations discussing rumours as a four-way classification task: supporting, denying, questioning and commenting. The performance of our best performing models have shown that automatically identifying stance is a task that can be successfully tackled using machine learning. Our results support and further advance earlier works on rumour stance classification (Zubiaga et al., 2016a; Lukasik et al., 2016). Naturally, performance of the models can and should be improved further in future work. Our experiments on automated rumour verification have also highlighted the importance of stance classification in the rumour resolution process, where stance can be used as one of the input features to rumour verification models or as an auxiliary task in multi-task learning set up. Furthermore, the task of stance classification can be applied beyond the context of rumours, for example to determine the side of a respondent in a debate. Our experiments have shown that our approach to rumour stance classification, that takes into account the sequence of responses, can also be applied for this task in broader settings.

RQ4 Which linguistic and network features are indicative of stance categories? We performed evaluation of two versions of tweet representations and the

effect of adding manually selected extra features representing various aspects of the data: local, relational, structural and social, on the model performance. As a result we identified a set of important features, that lead to improved model performance for PHEME and RumourEval 2017 datasets due to their relevance to the task as well as the dataset properties. The identification of a set of features that would be indicative of the stance of the users, participating in a conversation, across different datasets, is possible because expressions of support, denial and question are linguistically similar even when discussing various topics. However model performance still strongly depends on the quality of word-embedding model used for text representation and its vocabulary size. The identified feature set includes lexical features such as counts of swear and negation words. While negation words correlate strongly with the denying class they are also commonly present in posts from other classes. The model also makes use of punctuation features such as presence of exclamation marks, periods and question marks, with question marks being a strong indicator of questioning class. A strong indicator of the supporting class was a binary feature showing whether a tweet was a source of a conversation. Other features include tweet and content formatting as well as cosine similarity with other posts in the conversation.

RQ5 Can we leverage the sequence of responses to improve stance classification models? We have tested the importance of conversation context in identifying the stance of an individual post by comparing classification models that process posts individually against those working with pairs of posts or branches of posts. We have shown that sequential models, which take into account branches of posts outperform models utilising only single posts or pairs of posts. Thus we can conclude that the sequence of responses is indeed an important piece of information for the models classifying the stance towards rumours and should therefore be employed.

8.1.2 Rumour veracity classification

RQ1 Can we predict rumour veracity from social media conversations? We have studied automated rumour verification of Twitter conversation trees as a classification task with either true, false or unverified outcomes. We have developed and compared several methodological approaches to the task: two non-neural models (SVM and RF), which mainly focuses on the source tweet conveying the rumour, a neural sequential approach that models branches composed of the source tweet and the responses following it and a multi-task learning approach that allows to leverage

the relatedness between the tasks in rumour verification pipeline in order to improve veracity classification performance.

The outcomes of our experiments have shown the potential of machine learning models for rumour verification but they have also highlighted the difficulty of the task when pursuing a realistic evaluation scenario, which arises from the fact that each of the rumours covers new topics, with different vocabulary and users participating in discussions, which also leads to strong variation in proportion of true and false rumours between events.

RQ2 Which aspects of a conversation discussing a rumour are helpful for resolving rumour veracity? Are patterns of support and denial in a conversation indicative of its veracity? We have shown that a neural approach modelling a sequence of inputs performs better than non-sequential, non-neural approaches (SVM and RF models) in the majority of experiments we have performed. However this is not a universal conclusion in all datasets and feature combinations, as in some cases SVM outperforms *branch-LSTM*, however in the majority of cases *branch-LSTM* performs better.

We performed extensive experiments investigating the effects of various feature sets into rumour verification models. We were able to successfully identify features which benefit specific models on each of the datasets. However, as many of the feature sets benefited some of the models on one or more of the datasets, but not consistently across models and datasets, this does not lead to a definitive answer as to which features are the most indicative of rumour veracity overall. This highlights the need to test models developed in the future on multiple different dataset using realistic leave-one-event-out set up.

We have shown that the stance of a response towards a rumour improves classification performance of both sequential (*branch-LSTM* on the full PHEME dataset) and non-sequential (*NileTMRG* (Enayet and El-Beltagy, 2017) and *NileTMRG** on RumourEval 2017) models, therefore it has the potential of being a universal indicative feature.

RQ6 How can we leverage the interaction between the task of rumour verification and other tasks such as stance classification and rumour detection? We have proposed a rumour verification model that achieves improved performance for veracity classification by leveraging the task relatedness between rumour detection, rumour verification and stance classification, through a multi-task learning approach.

We have compared single task learning approaches with the proposed multi-task learning approaches that pair the verification task with the stance or rumour detection tasks, as well as using all tree tasks in one model. Our results show that the joint learning of two tasks from the verification pipeline outperforms a single-learning approach to rumour verification. Furthermore, the combination of all three tasks leads to additional performance improvements.

RQ7 How can we define and use model uncertainty in rumour verification? What does the uncertainty of predictions tell us about the task or the dataset? We have presented a method for obtaining model and data uncertainty estimates on the task of rumour verification in Twitter conversations.

We have demonstrated two ways in which uncertainty estimates can be used to remove instances that are likely to be incorrectly predicted so that making a decision concerning those instances can be passed to a human. The effects of data uncertainty and model uncertainty vary across datasets due to differences in their properties. The methods presented can be chosen and tuned using knowledge of the properties of the data at hand.

We presented a way to use uncertainty estimates to trace and interpret model decisions over time. We do not find that levels of uncertainty decrease over time in all of the cases, neither do we find a strong correlation between uncertainty and conversation size. We believe this can be due to the fact that every new response can add information of a different stance and quality, hence it can affect the model by either increasing or decreasing uncertainty.

We also find the relation between uncertainty levels and class labels to be dataset-specific, with the rumour instances with veracity value true having significantly lower levels of uncertainty across all datasets. This could be due to the prominent proportion of this class in all of the datasets.

8.2 Directions for future research

There are a few directions in which future work can focus. In this final section, we outline some of the major directions, based on the tasks that were tackled in this thesis.

8.2.1 Rumour stance classification

Stance classification is a challenging task that can be applied beyond the domain of rumours. While we proposed stance classification models that achieve good per-

formance, there is still a scope for improvement through addressing the following aspects of the problem. In our experiments we have highlighted that strong class imbalance towards commenting tweets that are not concerned with the veracity of a rumour is a challenge and we believe that the imbalance will also be prominent in other realistic datasets that were not manually balanced. Therefore we believe it is important to tackle this issue in future work through testing techniques such as cost-sensitive learning and data augmentation.

Representation learning aspect is also important. We have observed that models which have outperformed *branch-LSTM* in the RumourEval 2019 competition setting have used more recent language representation models, namely ELMO (Peters et al., 2018) and BERT (Devlin et al., 2018) in their approaches. These novel language models, pre-trained on large corpora, have wide vocabulary coverage, which is important for generalisation especially when training and fine-tuning on small datasets, such as the stance datasets that we work with here. However there is a need for further studies investigating the way of incorporating these into stance classification models while also taking into account the context provided by the conversation.

Judging by the improvement in performance of the tree CRF over the linear CRF model and of the branch-LSTM model over the tree CRF, there is potential in pursuing modeling the tree conversation structure using a neural approach. While there are recent works using tree-structured models processing tree-like rumour conversations (Ma et al., 2018a), they are concerned with the identification of rumour veracity rather than stance of each of the responses. However approaches with complex neural architectures usually are computationally intensive and prone to overfitting on small datasets. Therefore further investigation on the benefits of modelling tree conversation structure with neural approaches is required.

8.2.2 Rumour veracity classification

In the introduction to this thesis (chapter 1) we discussed the qualities that a good rumour verification system should have, and addressed some of them in this work. Here we would like to return to these requirements and suggest our vision for future work.

The verification system should be accurate, and even given recent advances there is still a need for improvement. Information provided in a single post is not usually enough to verify a rumour and there is a need in adding further knowledge into the models. Such knowledge can be provided by incorporating full tree-like conversation structures of parallel conversations as well as articles and web-pages,

linked to the tweets, or discussing the same rumours. It also can be in the form of background knowledge that 'explains' some concepts to the model that are 'common sense' to human. For example, one could use a database containing relations between countries and their capitals, incorporate this knowledge in the model, such that it is able to make an inference that if something happened in the capital it also means it happened in that country.

A multitask learning approach to rumour verification can be potentially improved by adapting the training schedule to account for different dataset sizes, such that none of the tasks dominates the model and by incorporating the hierarchy between tasks into the model. Leveraging advances in language modelling and representation learning is also important, as highlighted above for the stance classification task.

Another requirement for the verification model is that it should be able to generalise to unseen rumours. In order to pursue that goal we have used leave-one-event-out cross-validation. We believe that this practice should be widely adopted as it imitates a realistic scenario of new unforeseen events. This set up, however, is very challenging. To achieve good results large annotated datasets are required, thus revealing a need to find ways to create them fast and in less expensive ways, as currently high quality is guaranteed by manual labour of journalist professional. Additionally, state-of-the-art language models that were pre-trained on large amounts of text should be utilised and helpful for model generalisation abilities. Also, as rumours can cover any topics, from politics to crisis events to celebrity gossip, it may be wise to identify several prominent types and then train models for each specific type to provide models with appropriate inductive bias.

In this work we have addressed the need for providing models' level of uncertainty to its user and it has proven to be beneficial, and thus this line of work can profit from further investigations of other methods of uncertainty estimation as well as links between uncertainty values and linguistic features of the input. Developed models should utilise all of the information available at that point in time and adjust its predictions and uncertainty in the light of new information.

Another goal for future verification models is the ability to resolve rumours at an early stage, so as to prevent propagation and its subsequent harmful effects. As highlighted by other studies in the Related work section 2.2, models have to find ways to utilise the information and features that are available soon after the rumour starts.

We believe that in future work there should be more focus on a model's ability to provide justification and/or explanations for its predictions. There are

multiple paths through which this might be pursued. One could try to understand the model's behaviour better through the use of neural attention (in the case of deep learning model) or use model explanation algorithms such as LIME (Ribeiro et al., 2016) or SHAP (Lundberg and Lee, 2017). By using local estimations the latter provide their judgement on contributions of each of the features. The use of adversarial examples could help highlight the weak points of the model. Another path to generating explanations is through the use of summarisation approaches, which, however would require datasets annotated accordingly.

Finally, a model's behaviour should not exhibit significant biases. There has been precedent of language models being biased and an ongoing discussion of their solutions (Tatman, 2017; Zhao et al., 2018). In the case of rumour verification models one of the strong biases is the source bias. The way the source should be treated is an ongoing discussion: whether all stories released by a generally trusted source should be trusted, and, equally, if a new source or a less reputable source publishes a story, whether it should be considered false. However, we can not deny that the source is a highly indicative feature in the machine learning models, but it should be incorporated with caution. This certainly poses an important line of future work in the rumour verification domain.

Bibliography

Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762, 2014.

Saif M Mohammad. # emotional tweets. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 246–255. Association for Computational Linguistics, 2012.

Saif Mohammad, Cody Dunne, and Bonnie Dorr. Generating high-coverage semantic orientation lexicons from overtly marked words and a thesaurus. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 599–608. Association for Computational Linguistics, 2009.

Finn Årup Nielsen. A new anew: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903*, 2011.

Xiaodan Zhu, Svetlana Kiritchenko, and Saif Mohammad. Nrc-canada-2014: Recent improvements in the sentiment analysis of tweets. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 443–447, 2014.

Elena Kochkina, Maria Liakata, and Isabelle Augenstein. Turing at semeval-2017 task 8: Sequential approach to rumour stance classification with branch-lstm. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 475–480, 2017.

Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, Michal Lukasik, Kalina Bontcheva, Trevor Cohn, and Isabelle Augenstein. Discourse-aware rumour stance classification in social media using sequential classifiers. *Information Processing & Management*, 54(2):273–290, 2018a.

Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. All-in-one: Multi-task learning for rumour verification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3402–3413, 2018.

Genevieve Gorrell, Elena Kochkina, Maria Liakata, Ahmet Aker, Arkaitz Zubiaga, Kalina Bontcheva, and Leon Derczynski. Semeval-2019 task 7: Rumoureval, determining rumour veracity and support for rumours. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 845–854, 2019.

Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, and Michal Lukasik. Stance classification in rumours as a sequential task exploiting the tree structure of social media conversations. In *Proceedings of COLING, the International Conference on Computational Linguistics*, 2016a.

Zhe Zhao, Paul Resnick, and Qiaozhu Mei. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1395–1405. ACM, 2015.

Elisa Shearer and Jeffrey Gottfried. News use across social media platforms 2017. *Pew Research Center*, 7, 2017.

Amanda Lee Hughes and Leysia Palen. Twitter adoption and use in mass convergence and emergency events. *International journal of emergency management*, 6 (3-4):248–260, 2009.

Sarah Vieweg, Amanda L Hughes, Kate Starbird, and Leysia Palen. Microblogging during two natural hazards events: what Twitter may contribute to situational awareness. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1079–1088. ACM, 2010.

Paul Earle, Michelle Guy, Richard Buckmaster, Chris Ostrum, Scott Horvath, and Amy Vaughan. Omg earthquake! can Twitter improve earthquake response? *Seismological Research Letters*, 81(2):246–251, 2010.

GW Allport and L Postman. The psychology of rumor., 1965.

Onook Oh, Manish Agrawal, and H Raghav Rao. Community intelligence and social media services: A rumor theoretic analysis of tweets during social crises. *Mis Quarterly*, pages 407–426, 2013.

Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. Detection and resolution of rumours in social media: A survey. *ACM Computing Surveys (CSUR)*, 51(2):32, 2018b.

Nic Newman, Richard Fletcher, Antonis Kalogeropoulos, and Rasmus Kleis Nielsen. Reuters institute digital news report 2019. 2019.

Jeremy Wright and Sajid David. Online harms white paper. *UK Government, April 2019, https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/793360/Online_Harms_White_Paper.pdf*, 2019.

Arthur L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3:210–229, 1959.

Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.

Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. *arXiv e-prints*, art. arXiv:1907.10121, Jul 2019.

Mark Aronoff and Janie Rees-Miller. *The handbook of linguistics*, volume 460. Wiley Online Library, 2001.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751, 2013c.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185, 2014.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*, pages 238–247, 2014.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(4):694–707, 2016.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 52nd Annual Meeting of the Association for Computational , 2014.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. Technical report, Technical report, OpenAI, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

James W Pennebaker, Martha E Francis, and Roger J Booth. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001): 2001, 2001.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *In the Proceedings of ICLR.*, 2019.

Jan Salomon Cramer. The origins of logistic regression. *Tinbergen Institute Working Paper*, 2002.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

FELIX GERS. *Long Short-Term Memory in Recurrent Neural Networks*. PhD thesis, Universität Hannover, 2001.

Alex Graves. Supervised sequence labelling with recurrent neural networks. 2012. *http://books.google.com/books*, 2012.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2342–2350, 2015.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681, 1997.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136, 2011.

Twan van Laarhoven. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Vahed Qazvinian, Emily Rosengren, Dragomir R Radev, and Qiaozhu Mei. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1589–1599. Association for Computational Linguistics, 2011.

Gang Liang, Wenbo He, Chun Xu, Liangyin Chen, and Jinquan Zeng. Rumor identification in microblogging systems based on users behavior. *IEEE Transactions on Computational Social Systems*, 2(3):99–108, 2015.

Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. Detection and resolution of rumours in social media: A survey. *ACM Comput. Surv.*, 51(2):32:1–32:36, February 2018c. ISSN 0360-0300. doi: 10.1145/3161603.

Warren A Peterson and Noel P Gist. Rumor and public opinion. *American Journal of Sociology*, 57(2):159–167, 1951.

Nicholas DiFonzo and Prashant Bordia. Rumor, gossip and urban legends. *Diogenes*, 54(1):19–35, 2007.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1):22–36, 2017.

Guoyong Cai, Hao Wu, and Rui Lv. Rumors detection in chinese via crowd responses. In *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on*, pages 912–917. IEEE, 2014.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J Jansen, Kam-Fai Wong, and Meeyoung Cha. Detecting rumors from microblogs with recurrent neural networks. In *IJCAI*, pages 3818–3824, 2016.

Jing Ma, Wei Gao, and Kam-Fai Wong. Detect rumors in microblog posts using propagation structure via kernel learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 708–717, 2017.

Jing Ma, Wei Gao, and Kam-Fai Wong. Rumor detection on Twitter with tree-structured recursive neural networks. In *Proceedings of the 56th Annual Meeting*

of the Association for Computational Linguistics (Volume 1: Long Papers), volume 1, pages 1980–1989, 2018a.

Quanzhi Li, Qiong Zhang, and Luo Si. Rumor detection by exploiting user credibility information, attention and multi-task learning. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 1173–1179, 2019a.

Theodore Caplow. Rumors in war. *Social Forces*, pages 298–302, 1947.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. Information credibility on Twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684. ACM, 2011.

Marcelo Mendoza, Bárbara Poblete Labra, and Carlos Castillo Ocaranza. Predicting information credibility in time-sensitive social media. *Emerald Group Publishing*, 2013.

Aditi Gupta and Ponnurangam Kumaraguru. Credibility ranking of tweets during high impact events. In *Proceedings of the 1st Workshop on Privacy and Security in Online Social Media*, page 2. ACM, 2012.

BJ Fogg and Hsiang Tseng. The elements of computer credibility. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 80–87. ACM, 1999.

John O'Donovan, Byungkyu Kang, Greg Meyer, Tobias Höllerer, and Sibel Adalii. Credibility in context: An analysis of feature distributions in Twitter. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 293–301. IEEE, 2012.

Kevin R Canini, Bongwon Suh, and Peter L Pirolli. Finding credible information sources in social networks based on content and social structure. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pages 1–8. IEEE, 2011.

James Thorne and Andreas Vlachos. Automated fact checking: Task formulations, methods and future directions. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3346–3359, 2018.

Craig Silverman. Verification handbook. *The European Journalism Centre (EJC)*, 2013.

Andreas Vlachos and Sebastian Riedel. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 18–22, 2014.

Ndapandula Nakashole and Tom M Mitchell. Language-aware truth assessment of fact candidates. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1009–1019, 2014.

Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. Computational fact checking from knowledge networks. *PloS one*, 10(6):e0128193, 2015.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. Semeval-2017 task 8: Rumoureval: Determining rumour veracity and support for rumours. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 69–76, 2017.

Robert H Knapp. A psychology of rumor. *Public opinion quarterly*, 8(1):22–37, 1944.

Nicholas DiFonzo, Prashant Bordia, and Ralph L Rosnow. Reining in rumors. *Organizational Dynamics*, 23(1):47–62, 1994.

Sardar Hamidian and Mona T Diab. Rumor detection and classification for Twitter data. In *Proceedings of the Fifth International Conference on Social Media Technologies, Communication, and Informatics (SOTICS)*, pages 71–77, 2015.

Sardar Hamidian and Mona Diab. Rumor identification and belief investigation on Twitter. In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 3–8, 2016.

Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang. Prominent features of rumor propagation in online social media. In *2013 IEEE 13th International Conference on Data Mining*, pages 1103–1108. IEEE, 2013.

Sejeong Kwon and Meeyoung Cha. Modeling bursty temporal pattern of rumors. In *Eighth International AAAI Conference on Weblogs and Social Media*, 2014.

Yumeng Qin, Dominik Wurzer, Victor Lavrenko, and Cunchen Tang. Spotting rumors via novelty detection. *arXiv preprint arXiv:1611.06322*, 2016.

Arkaitz Zubiaga, Maria Liakata, and Rob Procter. Exploiting context for rumour detection in social media. In *International Conference on Social Informatics*, pages 109–123. Springer, 2017.

Laura Tolosi, Andrey Tagarev, and Georgi Georgiev. An analysis of event-agnostic features for rumour classification in Twitter. In *Tenth International AAAI Conference on Web and Social Media*, 2016.

Farzindar Atefeh and Wael Khreich. A survey of techniques for event detection in Twitter. *Computational Intelligence*, 31(1):132–164, 2015.

Xiaomo Liu, Quanzhi Li, Armineh Nourbakhsh, Rui Fang, Merine Thomas, Kajsa Anderson, Russ Kociuba, Mark Vedder, Steven Pomerville, Ramdev Wudali, et al. Reuters tracer: A large scale system of detecting & verifying real-time news events from Twitter. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 207–216. ACM, 2016.

Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Sameena Shah, Robert Martin, and John Duprey. Reuters tracer: Toward automated news production using large scale social media data. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1483–1493. IEEE, 2017.

Michael Mathioudakis and Nick Koudas. Twittermonitor: trend detection over the Twitter stream. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 1155–1158. ACM, 2010.

Preslav Nakov, Alberto Barrón-Cedeño, Tamer Elsayed, Reem Suwaileh, Lluís Màrquez, Wajdi Zaghouani, Pepa Atanasova, Spas Kyuchukov, and Giovanni Da San Martino. Overview of the clef-2018 checkthat! lab on automatic identification and verification of political claims. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 372–387. Springer, 2018.

Tamer Elsayed, Preslav Nakov, Alberto Barrón-Cedeño, Maram Hasanain, Reem Suwaileh, Giovanni Da San Martino, and Pepa Atanasova. Overview of the clef-2019 checkthat! lab: Automatic identification and verification of claims. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 301–321. Springer, 2019.

Saif M Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the International Workshop on Semantic Evaluation, SemEval*, volume 16, 2016.

Matt Thomas, Bo Pang, and Lillian Lee. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 327–335. Association for Computational Linguistics, 2006.

Sarvesh Ranade, Rajeev Sangal, and Radhika Mamidi. Stance classification in online debates by recognizing users' intentions. In *Proceedings of the SIGDIAL 2013 Conference*, pages 61–69, 2013.

Ju-han Chuang and Shukai Hsieh. Stance classification on ptt comments. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, 2015.

Gabriele Pergola, Lin Gui, and Yulan He. Tdam: A topic-dependent attention model for sentiment analysis. *Information Processing & Management*, 56(6):102084, 2019.

Bo Wang, Maria Liakata, Arkaitz Zubiaga, and Rob Procter. Tdparse: Multi-target-specific sentiment recognition on Twitter. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 483–493, 2017.

Saif M Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. Stance and sentiment in tweets. *ACM Transactions on Internet Technology (TOIT)*, 17(3):26, 2017.

Parinaz Sobhani, Saif Mohammad, and Svetlana Kiritchenko. Detecting stance in tweets and analyzing its interaction with sentiment. In *Proceedings of the fifth joint conference on lexical and computational semantics*, pages 159–169, 2016.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer, 2005.

Benjamin Riedel, Isabelle Augenstein, Georgios P Spithourakis, and Sebastian Riedel. A simple but tough-to-beat baseline for the fake news challenge stance detection task. *arXiv preprint arXiv:1707.03264*, 2017.

Andreas Hanselowski, PVS Avinesh, Benjamin Schiller, Felix Caspelherr, Debanjan Chaudhuri, Christian M Meyer, and Iryna Gurevych. A retrospective analysis of the fake news challenge stance-detection task. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1859–1874, 2018.

Piroska Lendvai and Uwe D Reichel. Contradiction detection for rumorous claims. *arXiv preprint arXiv:1611.02588*, 2016.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PloS one*, 11(3):e0150989, 2016b.

Marcelo Mendoza, Barbara Poblete, and Carlos Castillo. Twitter under crisis: Can we trust what we RT? In *1st Workshop on Social Media Analytics*, SOMA'10, pages 71–79, 2010.

Rob Procter, Farida Vis, and Alex Voss. Reading the riots on Twitter: methodological innovation for the analysis of big data. *International journal of social research methodology*, 16(3):197–214, 2013.

Anh Dang, Michael Smit, Abidalrahman Moh'd, Rosane Minghim, and Evangelos Milios. Toward understanding how users respond to rumours in social media. In *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*, pages 777–784. IEEE, 2016.

Cynthia Nichols, Lori Melton McKinnon, and Anna Geary. Rumor has it: examining the effects of facebook addiction on political knowledge gullibility. *The Journal of Social Media in Society*, 5(1):229–264, 2016.

Alessandro Balestrucci, Rocco De Nicola, Omar Inverso, and Catia Trubiani. Identification of credulous users on Twitter. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 2096–2103. ACM, 2019a.

Alessandro Balestrucci, Rocco De Nicola, Marinella Petrocchi, and Catia Trubiani. Do you really follow them? automatic detection of credulous Twitter users. *arXiv preprint arXiv:1909.03851*, 2019b.

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. Stance Detection with Bidirectional Conditional Encoding. In *Proceedings of EMNLP*, 2016.

Jiachen Du, Ruifeng Xu, Yulan He, and Lin Gui. Stance classification with target-specific neural attention networks. In *International Joint Conferences on Artificial Intelligence*, 2017.

Li Zeng, Kate Starbird, and Emma S Spiro. #unconfirmed: Classifying rumor stance in crisis-related social media messages. In *Tenth International AAAI Conference on Web and Social Media*, 2016.

Michal Lukasik, P. K. Srijith, Duy Vu, Kalina Bontcheva, Arkaitz Zubiaga, and Trevor Cohn. Hawkes processes for continuous time sequence classification: an application to rumour stance classification in Twitter. In *Proceedings of the 54th Meeting of the Association for Computational Linguistics*, pages 393–398. Association for Computer Linguistics, 2016. ISBN 978-1-945626-01-2.

Endang Wahyu Pamungkas, Valerio Basile, and Viviana Patti. Stance classification for rumour analysis in Twitter: Exploiting affective information and conversation structure. *arXiv preprint arXiv:1901.01911*, 2019.

Eunsoo Seo, Prasant Mohapatra, and Tarek Abdelzaher. Identifying rumors and their sources in social networks. In *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR III*, volume 8389, page 83891I. International Society for Optics and Photonics, 2012.

William Yang Wang. Liar, liar pants on fire: A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 422–426, 2017.

Kevin Roitero, Gianluca Demartini, Stefano Mizzaro, and Damiano Spina. How many truth levels? six? one hundred? even more? validating truthfulness of statements via crowdsourcing. In *Proceedings of the 2nd International Workshop on Rumours and Deception in Social Media*, 2018.

Cheng Chang, Yihong Zhang, Claudia Szabo, and Quan Z Sheng. Extreme user and political rumor detection on Twitter. In *International Conference on Advanced Data Mining and Applications*, pages 751–763. Springer, 2016.

Majed Alrubaian, Muhammad Al-Qurishi, Mabrook Al-Rakhami, and Atif Alamri. A credibility assessment model for online social network content. In *From Social Data Mining and Analysis to Prediction and Community Detection*, pages 61–77. Springer, 2017.

Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, and Sameena Shah. Real-time rumor debunking on Twitter. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1867–1870. ACM, 2015.

Ke Wu, Song Yang, and Kenny Q Zhu. False rumors detection on sina weibo by propagation structures. In *2015 IEEE 31st international conference on data engineering*, pages 651–662. IEEE, 2015.

Weiling Chen, Chai Kiat Yeo, Chiew Tong Lau, and Bu Sung Lee. Behavior deviation: An anomaly detection view of rumor preemption. In *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 1–7. IEEE, 2016.

Omar Enayet and Samhaa R El-Beltagy. Niletmrg at Semeval-2017 task 8: Determining rumour and veracity support for rumours on Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 470–474, 2017.

Quanzhi Li, Qiong Zhang, and Luo Si. eventai at semeval-2019 task 7: Rumor detection on social media by exploiting content, user credibility and propagation information. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 855–859, 2019b.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

Zili Zhang, Ziqiong Zhang, and Hengyun Li. Predictors of the authenticity of internet health rumours. *Health Information & Libraries Journal*, 32(3):195–205, 2015.

Alton Yeow Kuan Chua and Snehasish Banerjee. Linguistic predictors of rumor veracity on the internet. 2016.

Uwe D Reichel and Piroska Lendvai. Veracity computing from lexical cues and perceived certainty trends. *arXiv preprint arXiv:1611.02590*, 2016.

Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, page 13. ACM, 2012.

Soroush Vosoughi. *Automatic detection and verification of rumors on Twitter*. PhD thesis, Massachusetts Institute of Technology, 2015.

YeKang Yang, Kai Niu, and ZhiQiang He. Exploiting the topology property of social network for rumor detection. In *2015 12th International Joint Conference*

on Computer Science and Software Engineering (JCSSE), pages 41–46. IEEE, 2015.

Shihan Wang and Takao Terano. Detecting rumor patterns in streaming social media. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2709–2715. IEEE, 2015.

Sejeong Kwon, Meeyoung Cha, and Kyomin Jung. Rumor detection over varying time windows. *PloS one*, 12(1):e0168344, 2017.

Georgios Giasemidis, Colin Singleton, Ioannis Agrafiotis, Jason RC Nurse, Alan Pilgrim, Chris Willis, and Danica Vukadinovic Greetham. Determining the veracity of rumours on Twitter. In *International Conference on Social Informatics*, pages 185–205. Springer, 2016.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Tong Chen, Lin Wu, Xue Li, Jun Zhang, Hongzhi Yin, and Yang Wang. Call attention to rumors: Deep attention based recurrent neural networks for early rumor detection. *arXiv:1704.05973*, 2017.

Sebastian Dungs, Ahmet Aker, Norbert Fuhr, and Kalina Bontcheva. Can rumour stance alone predict veracity? In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3360–3370, 2018.

Sumeet Kumar and Kathleen M Carley. Tree lstms with convolution units to predict stance and rumor veracity in social media conversations. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 5047–5058, 2019.

Jing Ma, Wei Gao, and Kam-Fai Wong. Detect rumor and stance jointly by neural multi-task learning. In *Companion Proceedings of the The Web Conference 2018*, pages 585–593. International World Wide Web Conferences Steering Committee, 2018b.

Aditi Gupta, Hemank Lamba, Ponnurangam Kumaraguru, and Anupam Joshi. Faking sandy: characterizing and identifying fake images on Twitter during hurricane sandy. In *Proceedings of the 22nd international conference on World Wide Web*, pages 729–736. ACM, 2013.

Christina Boididou, Symeon Papadopoulos, Yiannis Kompatsiaris, Steve Schifferes, and Nic Newman. Challenges of computational verification in social multimedia. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 743–748. ACM, 2014.

Christina Boididou, Symeon Papadopoulos, Lazaros Apostolidis, and Yiannis Kompatsiaris. Learning to detect misleading content on Twitter. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pages 278–286. ACM, 2017.

J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

John G Cleary and Leonard E Trigg. K*: An instance-based learner using an entropic distance measure. In *Machine Learning Proceedings 1995*, pages 108–114. Elsevier, 1995.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Sara Rosenthal and Kathleen McKeown. I couldnt agree more: The role of conversational structure in agreement and disagreement detection in online discussions. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 168, 2015.

William Ferreira and Andreas Vlachos. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 1163–1168, 2016.

Alan G Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90, 1971.

Kalina Bontcheva, Leon Derczynski, Adam Funk, Mark A. Greenwood, Diana Maynard, and Niraj Aswani. TwitIE: An open-source information extraction pipeline for microblog text. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing, RANLP*, pages 83–90. Association for Computational Linguistics, 2013.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *Icml*, volume 97, page 35, 1997.

Yishi Zhang, Shujuan Li, Teng Wang, and Zigang Zhang. Divergence-based feature selection for separate classes. *Neurocomputing*, 101:32–42, 2013.

James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2011.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*, 2012.

Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Sren Kaae Sønderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, Jeffrey De Fauw, Michael Heilman, Diogo Moitinho de Almeida, Brian McFee, Hendrik Weideman, Gbor Takács, Peter de Rivaz, Jon Crall, Gregory Sanders, Kashif Rasul, Cong Liu, Geoffrey French, and Jonas Degrave. Lasagne: First release., August 2015. URL `http://dx.doi.org/10.5281/zenodo.27878`.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

James Bergstra, Daniel Yamins, and David D Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. *ICML (1)*, 28:115–123, 2013.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges*

for NLP Frameworks, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

Edward Loper and Steven Bird. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*, 2002.

Ruoyao Yang, Wanying Xie, Chunhua Liu, and Dong Yu. Blcu_nlp at semeval-2019 task 7: An inference chain-based gpt model for rumour evaluation. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 1090–1096, 2019.

François Chollet et al. Keras. `https://github.com/keras-team/keras`, 2015.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027, 2016a.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Andrew Moore and Paul Rayson. Bringing replication and reproduction together with generalisability in nlp: Three reproduction studies for target dependent sentiment analysis. *arXiv preprint arXiv:1806.05219*, 2018.

Michał Woźniak, Manuel Graña, and Emilio Corchado. A survey of multiple classifier systems as hybrid systems. *Information Fusion*, 16:3–17, 2014.

Alceu S Britto Jr, Robert Sabourin, and Luiz ES Oliveira. Dynamic selection of classifiersa comprehensive review. *Pattern Recognition*, 47(11):3665–3680, 2014.

Rich Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Sluice networks: Learning what to share between loosely related tasks. *stat*, 1050:23, 2017.

Yongxin Yang and Timothy M Hospedales. Trace norm regularised deep multi-task learning. *arXiv preprint arXiv:1606.04038*, 2016.

Yaser S Abu-Mostafa. Learning from hints in neural networks. *Journal of complexity*, 6(2):192–198, 1990.

Gustavo Aguilar, Suraj Maharjan, Adrian Pastor López Monroy, and Thamar Solorio. A multi-task approach for named entity recognition in social media data. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 148–153, 2017.

Man Lan, Jianxiang Wang, Yuanbin Wu, Zheng-Yu Niu, and Haifeng Wang. Multi-task attention-based neural networks for implicit discourse relationship representation and identification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1299–1308, 2017.

Héctor Martínez Alonso and Barbara Plank. When is multitask learning effective? semantic sequence prediction under varying data conditions. In *15th Conference of the European Chapter of the Association for Computational Linguistics*, 2017.

Joachim Bingel and Anders Søgaard. Identifying beneficial task relations for multi-task learning in deep neural networks. *arXiv preprint arXiv:1702.08303*, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on neural networks and learning systems*, 2019.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International conference on machine learning*, pages 1050–1059, 2016b.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems*, pages 7047–7058, 2018.

Yijun Xiao and William Yang Wang. Quantifying uncertainties in natural language processing tasks. *arXiv preprint arXiv:1811.07253*, 2018.

Li Dong, Chris Quirk, and Mirella Lapata. Confidence modeling for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 743–753, 2018.

Pierre-Antoine Jean, Sébastien Harispe, Sylvie Ranwez, Patrice Bellot, and Jacky Montmain. Uncertainty detection in natural language: a probabilistic model. In *Proceedings of the 6th International Conference on Web Intelligence, Mining and Semantics*, page 10. ACM, 2016.

Veronika Vincze. *Uncertainty detection in natural language texts*. PhD thesis, szte, 2015.

William H Kruskal and W Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

Alex Guy Kendall. *Geometry and uncertainty in deep learning for computer vision*. PhD thesis, University of Cambridge, 2019.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf`.

Rachael Tatman. Gender and dialect bias in youtubes automatic captions. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 53–59, 2017.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 15–20, 2018.

APPENDIX A

Features used in rumour verification experiments

Text

- *Word embeddings:* we represent each tweet as a vector with 300 dimensions, as the average of vector representations of the words in the tweet using word2vec (Mikolov et al., 2013a).

Attachments

We extract these features indicating presence of URLs or images as users attach URLs as well as photographic evidence or screenshots to show evidence or sources tu support the claims they are making.

- *Attachment of URL:* binary feature, indicating the presence of at least one URL in the tweet.
- *Attachment of an image:* binary feature, indicating the presence of at least one image attached to the tweet.

Lexicon

We have extract features that are indicating the presence of specific sets of words that are potentially indicative of rumour veracity or stance of responses.

- *Use of swear words:* number of 'bad' words present in a tweet. We use a list of 458 bad words[1]. These words might indicate strong emotions.
- *Use of negation:* number of negation words found in a tweet.

[1] `http://urbanoalvarez.es/blog/2008/04/04/bad-words-list/`

- *Use of wh- words:* number of wh- words found in a tweet. This is determined by looking at the presence of the following words: 'what', 'when', 'where', 'which', 'who', 'whom', 'whose', 'why', 'how'.

- *Use of words 'rumour' and 'unconfirmed':* binary features, indicating the presence of words 'rumour' and 'unconfirmed'.

- *Use of synonyms of 'false':* number of synonyms of 'false' found in a tweet. This is determined by looking at the presence of the following words: 'false', 'bogus', 'deceitful', 'dishonest', 'distorted', 'erroneous', 'fake', 'fanciful', 'faulty', 'fictitious', 'fraudulent', 'improper', 'inaccurate', 'incorrect', 'invalid', 'misleading', 'mistaken', 'phony', 'specious', 'spurious', 'unfounded', 'unreal', 'untrue', 'untruthful', 'apocryphal', 'beguiling', 'casuistic', 'concocted', 'cooked-up', 'counterfactual', 'deceiving', 'delusive', 'ersatz', 'fallacious', 'fishy', 'illusive', 'imaginary', 'inexact', 'lying', 'mendacious', 'misrepresentative', 'off the mark', 'sham', 'sophistical', 'trumped up', 'unsound'.

- *Use of antonyms of 'false':* number of antonyms of 'false' found in a tweet. This is determined by looking at the presence of the following words: 'accurate', 'authentic', 'correct', 'fair', 'faithful', 'frank', 'genuine', 'honest', 'moral', 'open', 'proven', 'real', 'right', 'sincere', 'sound', 'true', 'trustworthy', 'truthful', 'valid', 'actual', 'factual', 'just', 'known', 'precise', 'reliable', 'straight', 'substantiated'.

- *General lexicon features:* a vector of binary and count-based features taken from a collection of lexicons (Hu and Liu, 2004; Kiritchenko et al., 2014; Mohammad, 2012; Mohammad et al., 2009; Nielsen, 2011; Zhu et al., 2014).

Content formatting

Content formatting can be linked to rumour veracity as it might indicate the level of emotion (e.g. more emotional messages can be using more capital letters, more exclamation marks), or certainty of the author (question marks may identify uncertainty) etc.

- *Presence of question mark:* binary feature, indicating the presence or not of at least one question mark in the tweet.

- *Presence of exclamation mark:* binary feature, indicating the presence or not of at least one exclamation mark in the tweet.

- *Presence of period:* binary feature, indicating the presence or not of at least one period in the tweet.
- *Presence of hashtag:* binary feature, indicating the presence or not of at least one hashtag in the tweet.
- *Character count:* the length of the tweet in number of characters.
- *Word count:* the number of words in the tweet, counted as the number of space-separated tokens.
- *Ratio of capital letters:* number of capital letters in the tweet divided by character count.

User features

User features can give us hints pertaining to user's credibility and hence rumour veracity.

- *Number of followers:* count of users following the author of the post.
- *Follow ratio:* number of followers divided by a number of users the author of the post is following.
- *Account age:* number of days since account creation.
- *Statuses count:* number of tweets the account has posted.
- *Verified user:* binary feature, indicating whether the user has been verified by a platform.
- *User has URL:* binary feature, indicating whether the user's profile contains a URL.
- *Geolocation enabled:* binary feature, indicating whether user geolocation is enabled.

Social interactions

We study whether the number and types of social interactions, are indicative of rumour veracity.

- *Favourite count:* the number of times a tweet has been favourited.
- *Re-tweet count:* the number of times a tweet has been retweeted.

Stance

We use the stance labels predicted by our *branch-LSTM* classification model.

- *Proportion:* features showing ratio of supporting, denying and questioning tweets in the conversation tree (used in *SVM*, *RF* and *NileTMRG* models).
- *Labels:* stance labels of each of the tweets (used in sequential model).

Conversation tree

These features introduce information about the conversation around the tweet/rumour.

- *Similarity to the source tweet:* the cosine similarity between the word vector representation of the current tweet and the word vector representation of the source tweet (not used in *SVM*, *RF* and *NileTMRG* models as those models only use the source tweet).
- *Similarity to the other tweets in the conversation tree:* the cosine similarity score between the current tweet and the rest of the tweets in the conversation tree, excluding the tweets from the same author as that of the current tweet.

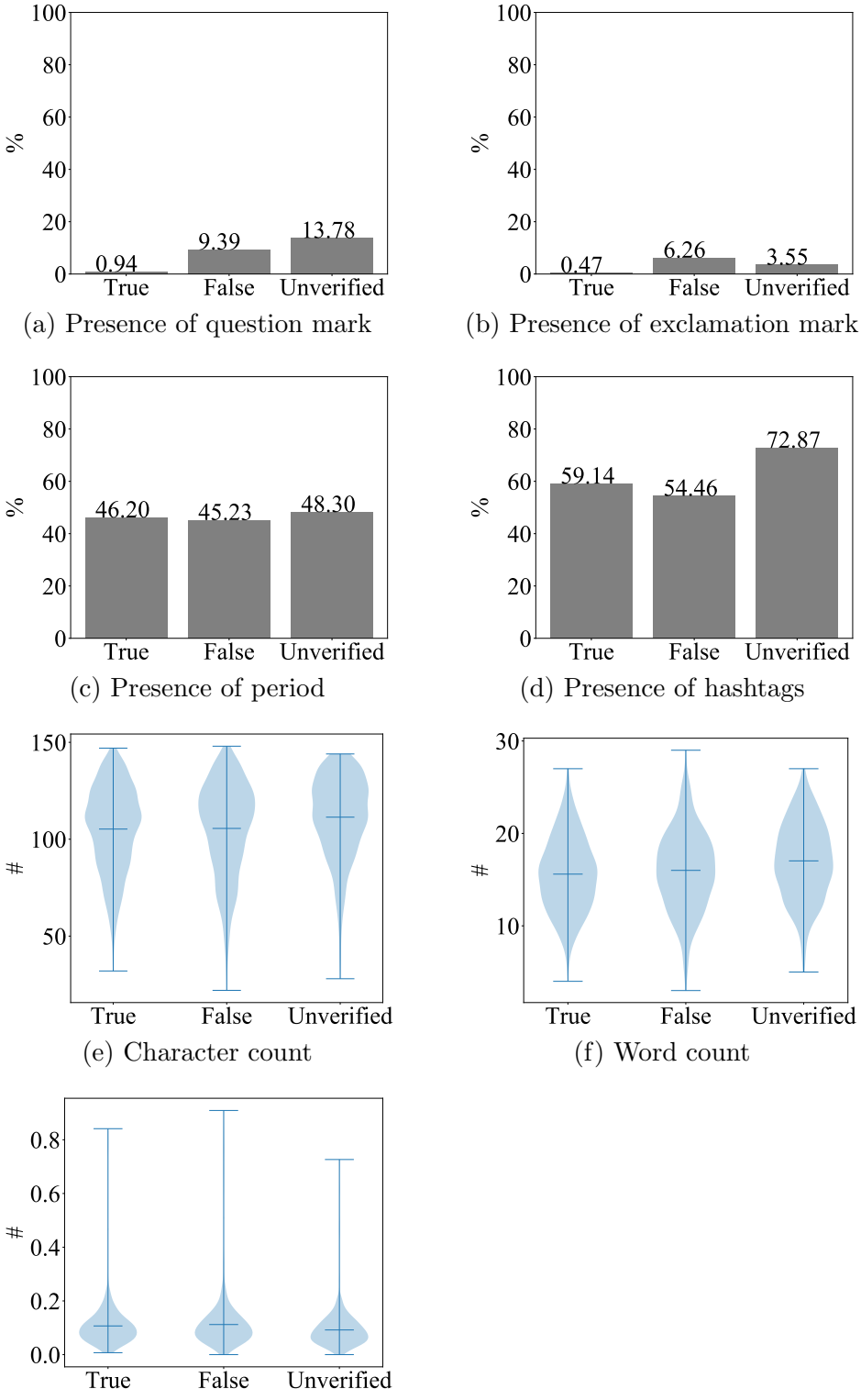
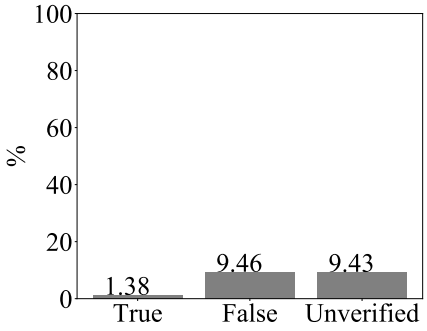
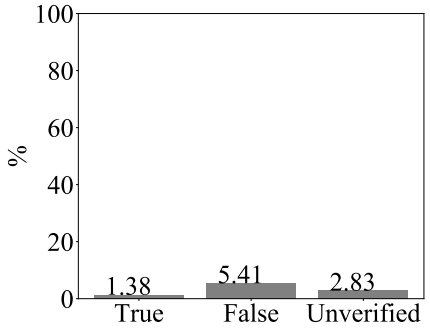


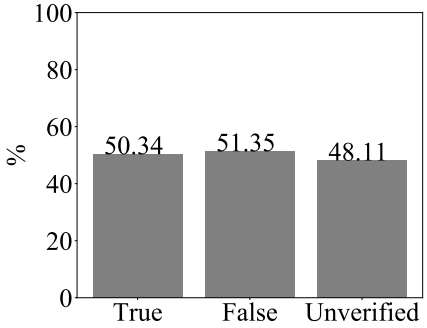
Figure A.1: Distribution of content formatting features per-class in source tweets of the PHEME dataset.



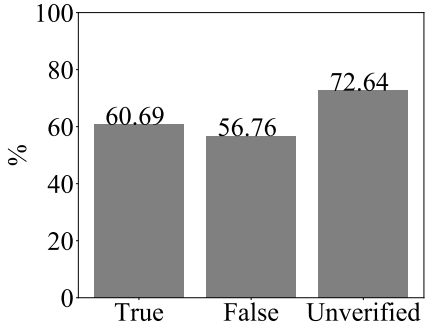
(a) Presence of question mark



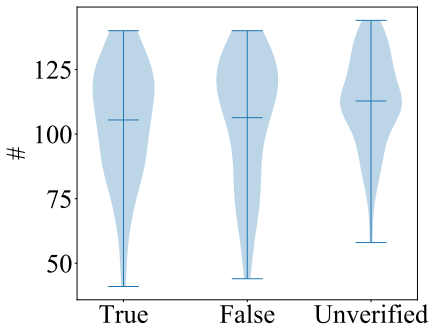
(b) Presence of exclamation mark



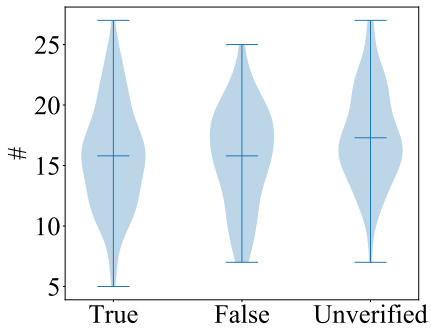
(c) Presence of period



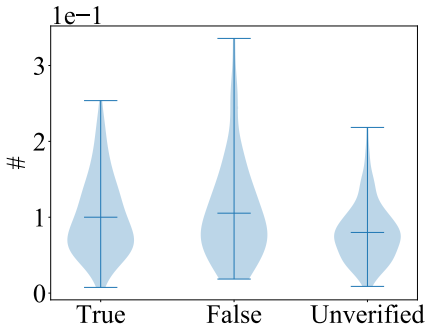
(d) Presence of hashtags



(e) Character count

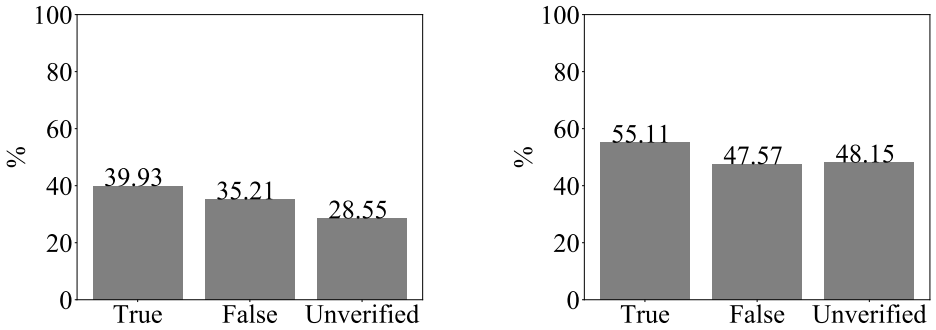


(f) Word count



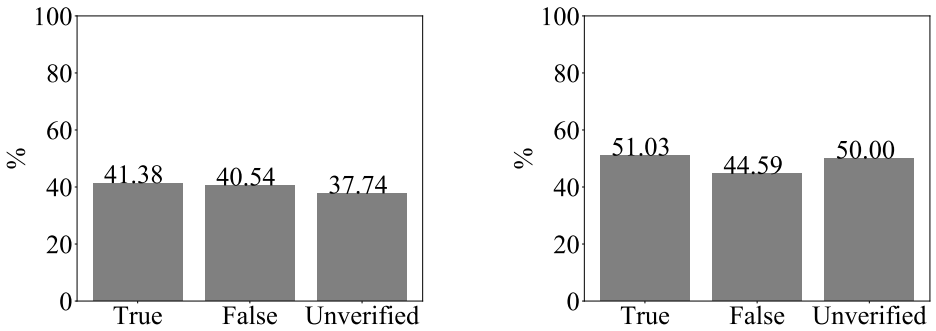
(g) Capital ratio

Figure A.2: Distribution of content formatting features per-class in source tweets of the RumourEval 2017 dataset.



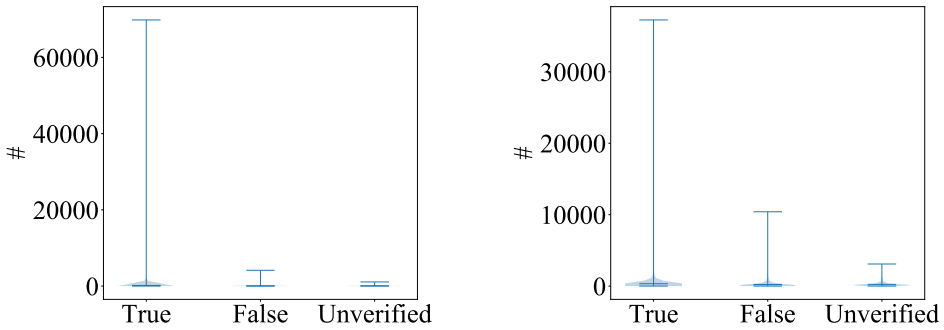
(a) Presence of image (b) Presence of URL

Figure A.3: Distribution of tweet attachment features per-class in source tweets of the PHEME dataset.



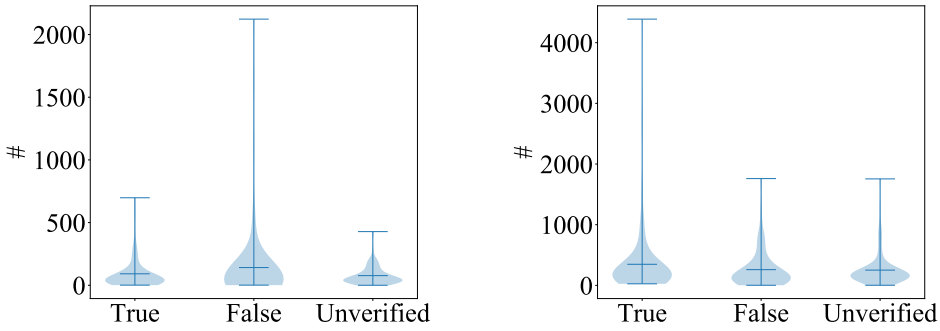
(a) Presence of image (b) Presence of URL

Figure A.4: Distribution of tweet attachment features per-class in source tweets of the RumourEval 2017 dataset.



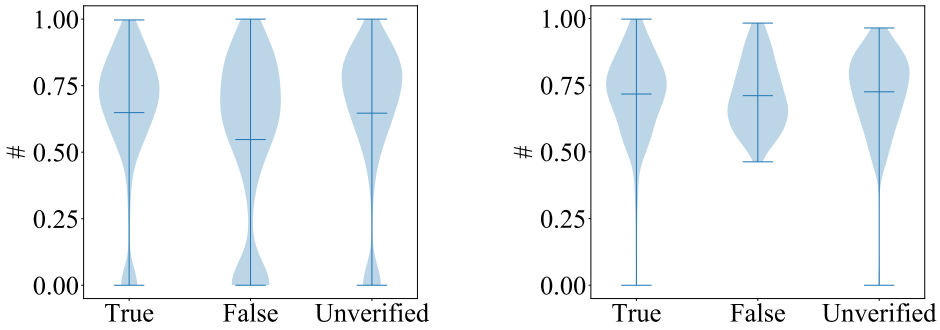
(a) Favourite count (b) Retweet count

Figure A.5: Distribution of social interaction features per-class in source tweets of the PHEME dataset.



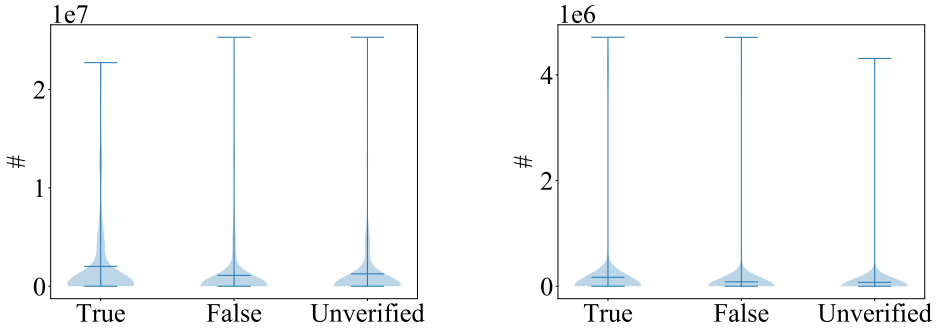
(a) Favourite count (b) Retweet count

Figure A.6: Distribution of social interaction features per-class in source tweets of the RumourEval 2017 dataset.



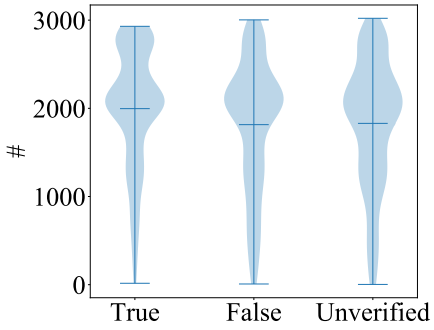
(a) PHEME dataset (b) RumourEval 2017

Figure A.7: Distribution of cosine similarity with respect to other tweets in the conversation tree feature per-class in source tweets.

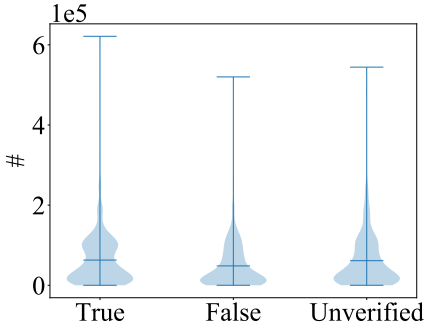


(a) Number of followers

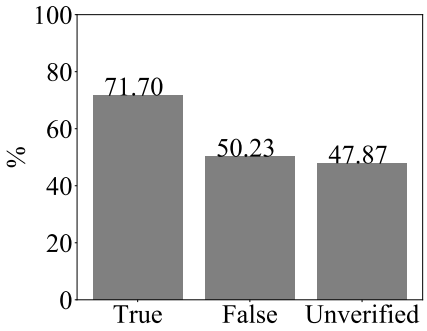
(b) Following to follower ratio



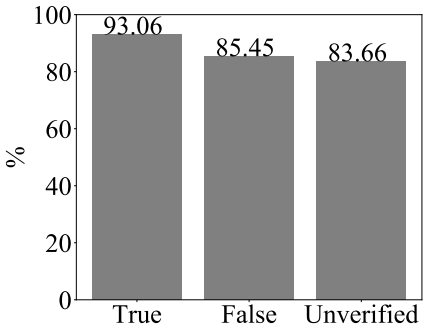
(c) Account age, days



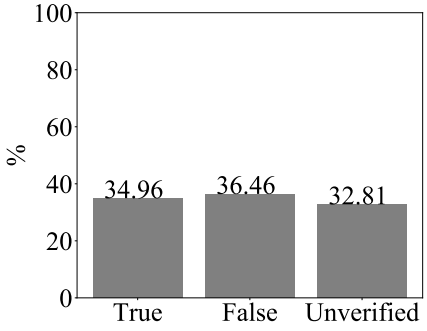
(d) Statuses count



(e) Verified user

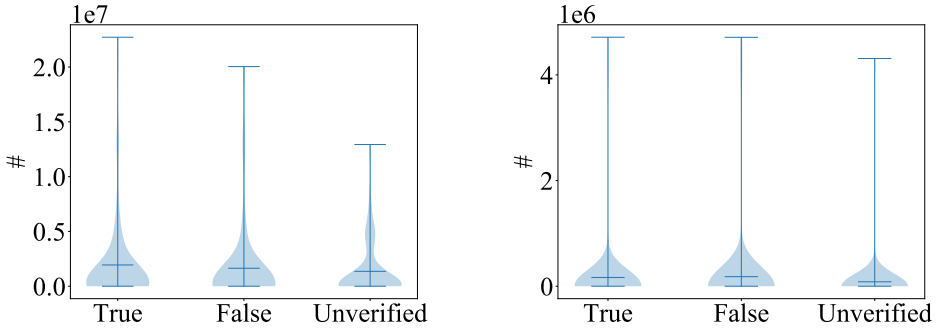


(f) User has URL



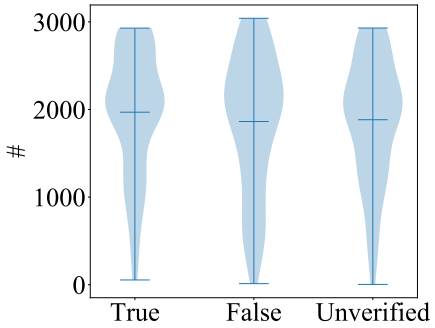
(g) Geolocation enabled

Figure A.8: Distribution of user features per-class in source tweets of the PHEME dataset.

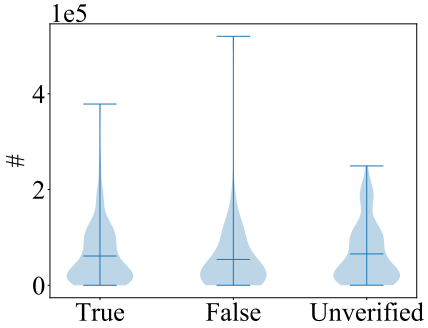


(a) Number of followers

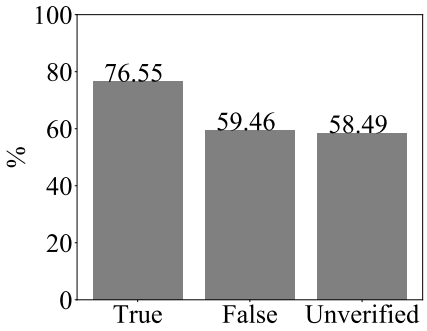
(b) Following to follower ratio



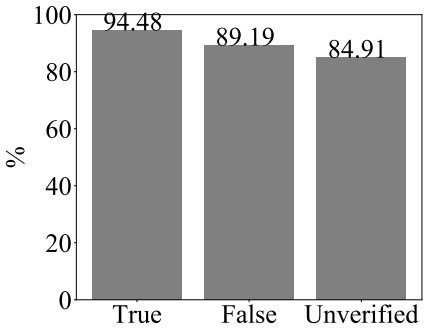
(c) Account age, days



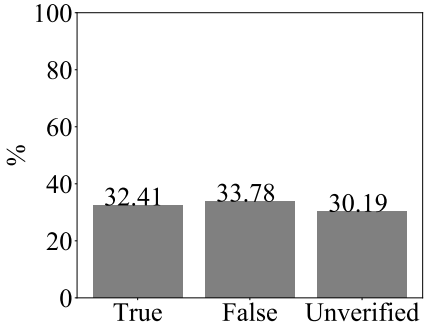
(d) Statuses count



(e) Verified user

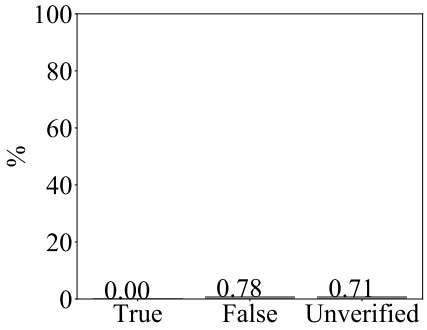


(f) User has URL

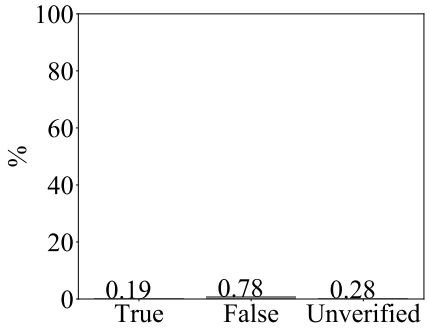


(g) Geolocation enabled

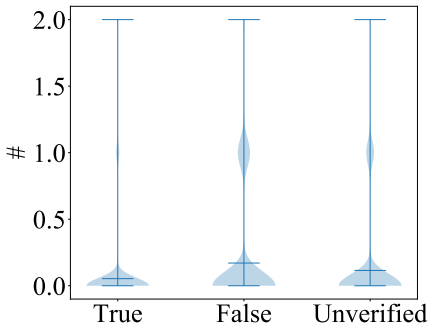
Figure A.9: Distribution of user features per-class in source tweets of the RumourEval 2017 dataset.



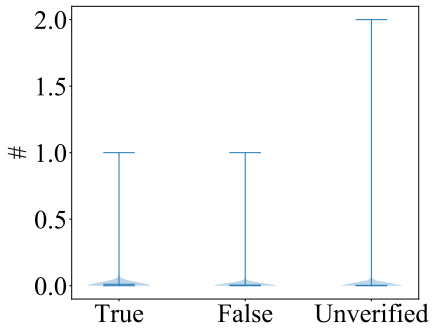
(a) Presence of 'rumour'



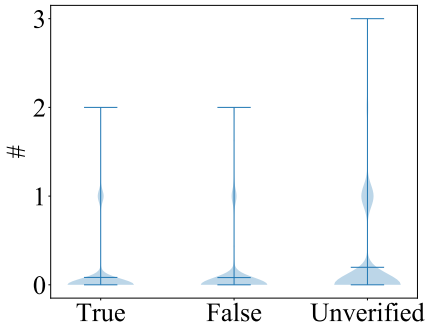
(b) Presence of 'unconfirmed'



(c) Use of negation

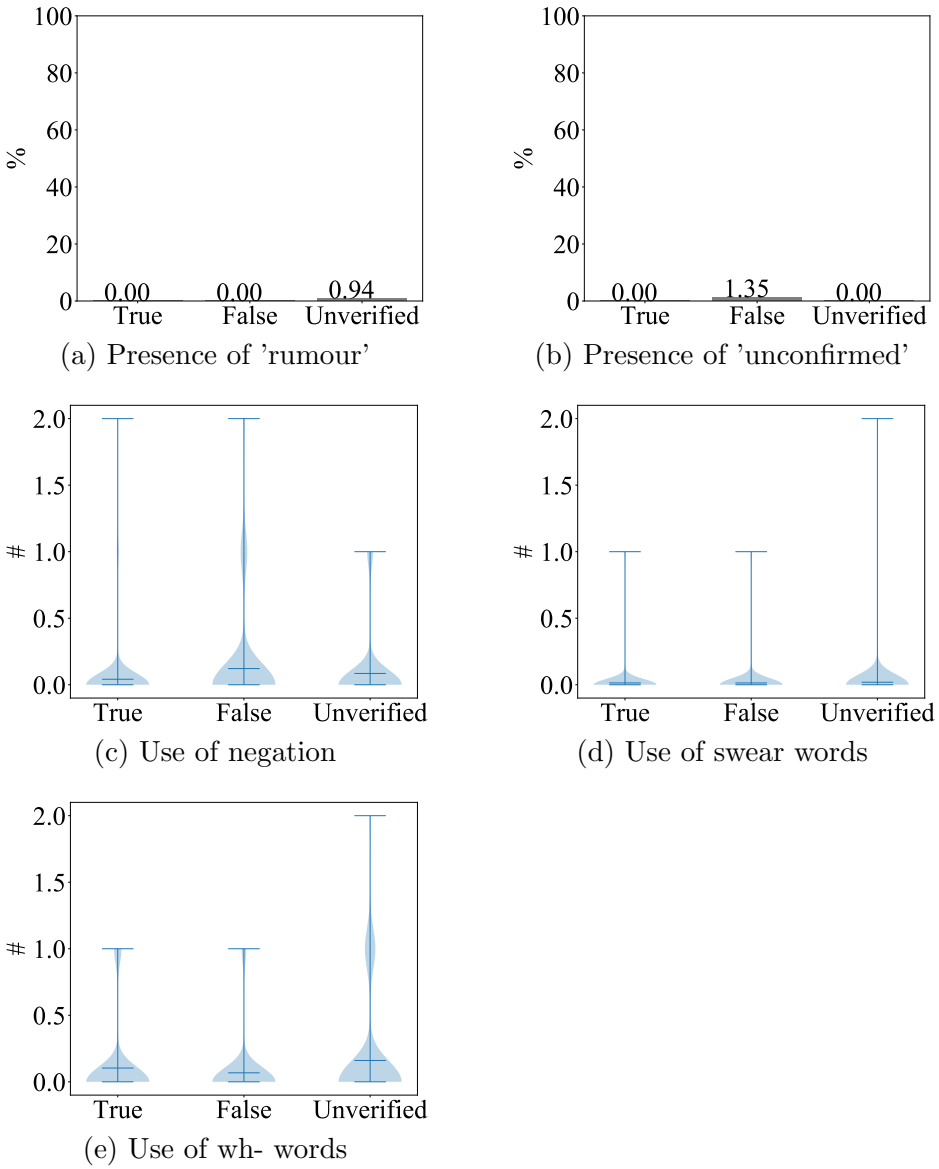


(d) Use of swear words



(e) Use of wh- words

Figure A.10: Distribution of lexicon features per-class in source tweets of the PHEME dataset.



(a) Presence of 'rumour'

(b) Presence of 'unconfirmed'

(c) Use of negation

(d) Use of swear words

(e) Use of wh- words

Figure A.11: Distribution of lexicon per-class in source tweets of RumourEval 2017 dataset